DATA-DRIVEN MATERIAL RECOGNITION AND PHOTOREALISTIC IMAGE EDITING USING DEEP CONVOLUTIONAL NEURAL NETWORKS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Paul Robert Upchurch

August 2018

DATA-DRIVEN MATERIAL RECOGNITION AND PHOTOREALISTIC IMAGE EDITING USING DEEP CONVOLUTIONAL NEURAL NETWORKS

Paul Robert Upchurch, Ph.D.

Cornell University 2018

Fully automatic processing of images is a key challenge for the 21st century. Our processing needs lie beyond just organizing photos by date and location. We need image analysis tools that can reason about photos like a human. For example, we need algorithms that can identify the people, activities, materials, illumination and 3D shapes depicted in photos. These *scene understanding* tasks are the key building blocks of many practical applications: automatic album creation & video summarization, intelligent sharing & privacy policies for social networks, automatic image captioning, visual question & answer, visual search & recommendation systems, acquisition of physical properties (e.g., reflectance and 3D shape), augmented reality object insertion, artistic stylization, smart photo manipulation, robot task planning and self-driving cars.

The goal of scene understanding is to infer a structured model of reality from a photo. This cannot be done perfectly because there can be many realities which produce the same image. Humans excel at using prior experience to guess the reality which underlies an image. A new approach to scene understanding has grown around the idea that computers can mimic human-level scene understanding by fitting an artificial neural network to labeled image data. This approach is called *deep learning* and it is dramatically more successful than previous approaches. Deep learning can be summarized in three key steps: (1) acquire millions of labeled images; (2) create a multi-layer convolutional neu-

ral network (aka deep CNN) model parameterized by millions of variables; and (3) adjust the parameters so that the CNN predicts the same as a human.

In this thesis we explore the three steps of deep learning through the lens of recognizing materials in a real-world scene and making structured changes to an image: we describe a practical method for efficiently gathering crowdsourced labels; a method for assigning a material label to every pixel of an image; a method for improving the accuracy of crowdsourced labels; and surprising new applications of CNNs for photorealistic image manipulation.

BIOGRAPHICAL SKETCH

Paul Upchurch was born in Austin, Texas, USA. His first introduction to computers was through his family. He fondly remembers building a Heathkit computer with his father. His first formal education in Computer Science was a summer course at Scripps College, Claremont, California as part of the Johns Hopkins Center for Talented Youth Program. After he completed the course material ahead of schedule the instructor told him to write his own programs. Students were prohibited from bringing video games to summer school so Paul solved this problem by writing a video game. This caused the computer lab to became a popular destination during recess.

While Paul continued his formal education at Stratford H.S. in Houston, Texas, he supplemented his high-school curriculum with Math classes at Rice University and summer Physics courses at Caltech in Pasadena, California. He regularly participated in high-school Science and Math contests at the local and state level.

In 1992, Paul entered the undergraduate program at Caltech and continued his formal education into Computer Science. He supplemented his coursework with practice of real-time 3D computer graphics and network protocols. After graduation, he went on to work in the computer industry in Silicon Valley, San Diego and Los Angeles. His work included network streaming video, console video games, multiplayer video games, and 3D visualization of space science missions. In 2012, Paul began his doctorate studies on computer vision and machine learning under the supervision of Kavita Bala, Noah Snaveley and Kilian Weinberger at Cornell University. After graduation, Paul's research at the intersection of computer vision, computer graphics and machine learning continues at Apple in Sunnyvale, California.

This dissertation is dedicated to my family.

ACKNOWLEDGEMENTS

I thank my advisor Prof. Kavita Bala for her insight, guidance and support throughout my time at Cornell. Her inexhaustible drive for excellence in herself and others is an inspiration. I thank my co-advisor Prof. Noah Snavely for his continual support, and advice. Kavita and Noah's insights on research and penetrating questions throughout my doctoral studies have been chiefly responsible for my growth as a researcher. I thank my committee member Prof. Kilian Q. Weinberger for our close collaboration and for his insights on machine learning. I thank my committee member Prof. Charles Van Loan and I also thank Prof. Robert Kleinberg for serving as a proxy on my committee. I thank Prof. Haym Hirsh for his insights and collaboration on crowdsourcing.

I thank my close collaborators for the enjoyable time and late nights we spent working together on research papers: Sean Bell, Jacob Gardner, Daniel Huaugge, Kevin Matzen, and Geoff Pleiss. I thank Kyle Wilson for his valuable help during a deadline rush. I also thank my collaborators and colleagues in the Cornell Graphics and Vision Lab and the Computer Science Department for many stimulating discussions of ideas and encouragement through shared interests in research: Sean Bell, Kevin Matzen, Daniel Hauagge, Balázs Kovács, Kyle Wilson, Scott Wehrwein, Jacob Gardner, Matt Kusner, Geoff Pleiss, Hubert Lin, Utkarsh Mall, Eston Schweickart, Ivaylo Boyadzhiev, Fujun Luan, Daniel Sedra, Jason Yosinski, and Ransen Niu. I thank the colleagues that I met at conferences, workshops and seminars for the exciting exchange of ideas and questions about research.

I thank Prof. Mathieu Desbrun, Prof. Peter Schröder and Kevin Hussey for their early sponsorship and creating a path for me to pursue academic research. I thank my undergraduate Professors for introducing me to the formal world of

Computer Science.

I thank my parents for continually supporting my education and work throughout my life. I thank my father Ed Upchurch for our many discussions of computing technology and my mother Anh Upchurch for her wisdom. I thank my brother Sean Upchurch for inspiring me to work harder by being a role-model of excellence. I thank my wife Chen Feng Ng for her support during my studies, for discussions about research, and for taking leaves from work to be with me in Ithaca.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Photos are one of the most important forms of data because they are rich in information yet easy to acquire, store, transmit and manipulate. In 2017, Google revealed that 1.2 billion photos are uploaded to their photo service every day [90]. There are over a thousand artificial satellites [96] of which over a hundred are dedicated to Earth observation [144]. Robots and self-driving cars routinely use cameras to sense the world. These open-ended sources of data have grown beyond our capacity for manual interpretation. Fully automatic processing of images is a key challenge for the 21st century. Cameras already augment photos with direct observables such as date and GPS coordinates. The next step is *scene understanding*, which extracts meaningful hidden information from a photo. Scene understanding comes in many forms (e.g., categorization of materials, objects and activities; identification of people; and extraction of illumination and 3D shape). Figure 1.1 illustrates categorization of materials and objects. Scene understanding methods are the key building blocks of many practical applications: automatic album creation & video summarization, intelligent sharing & privacy policies for social networks, automatic image captioning, visual question & answer, visual search & recommendation systems, acquisition of physical properties (e.g., reflectance and 3D shape), augmented reality object insertion, artistic stylization, smart photo manipulation, robot task planning and self-driving cars.

Scene understanding is a structured task—a photo is not unstructured noise but rather an image of reality. The goal of scene understanding is to infer a structured model of reality from a photo. This is a difficult problem called the

Figure 1.1: **Scene Understanding.** Reasoning about a scene (left) as a composition of categorical parts, for example materials and objects (right), is one form of scene understanding.

inverse rendering problem. Inverse rendering cannot be solved perfectly because there can be many realities which produce the same image. For example, in Figure 1.2 an artist has printed a warped picture of a toy car on paper which is indistinguishable from a real toy car at a certain vantage point. Notwithstanding optical illusions, humans generally excel at inverse rendering. Based on prior experience we can guess a likely reality which explains an image. A new approach to scene understanding has grown around the idea that computers can mimic human-level scene understanding by fitting an artificial neural network to labeled image data. This approach is called *deep learning* and it is dramatically more successful than previous approaches. Deep learning can be summarized in three key steps: (1) acquire millions of images with human-assigned labels for a scene understanding task; (2) create a multi-layer convolutional neural network (aka deep CNN) model parameterized by millions of variables; and (3) adjust the parameters (i.e., train the model) so that the CNN closely predicts the same labels given by humans. A trained CNN can achieve near-human

Figure 1.2: **Anamorphic illusion.** What material is this car (left) made of? Answering this question is difficult because there are multiple hypothetical realities consistent with the photograph. One hypothesis is that it is a metal toy car. Another hypothesis is that it is a warped drawing of a toy car printed on paper, lying flat on the surface of the table. When the object is touched (center) and rotated to be seen at a different angle (right) the correct reality is revealed. Credit: Philipp Hebold[2].

level performance on tasks previously thought to be intractable. Surprisingly, a trained CNN can be repurposed and achieve high-performance on a different task than the one for which the labels were acquired. This amazing fact is due to the common structured reality which underlies all images.

The road to high-performance CNNs is a long one with some questions answered and many still unanswered: what is the best way to collect large-scale image annotations? what kind of annotations are most effective for training a CNN? how accurate are human annotations? how do we turn human annotations into training labels? what is the best architecture for a CNN? how do we train a CNN? how do we repurpose a CNN? and which scene understanding tasks are crucial for creating models which can be effectively repurposed? In this thesis we explore some of these questions and discover a practical method for efficiently gathering crowdsourced labels (Chapter 2), a method for assigning a material label to every pixel of an image (Chapter 2), a method for improving the accuracy of crowdsourced labels (Chapter 3) and surprising new

[2] https://www.youtube.com/watch?v=o8Vs6jWVQZQ

applications of CNNs for photorealistic image manipulation (Chapter 4).

Chapter 2 presents the open, large-scale *Materials in Context Database* (MINC) and a state-of-the-art method for recognizing materials in the wild. Materials are important because they determine the appearance of everything in the world. Computer Graphics models reality as a collection of 3D surfaces & volumes made out of materials which interact with light. A material is a functional description of how light reflects and scatters from a surface or within a volume. Materials are a key concept for image rendering and inverse rendering. Furthermore, recognizing material classes (e.g., metal, paper, water) answers important questions about physical behaviour (will it melt? can I pick it up? can I stand on it?). Recognizing materials in real-world images is a challenging task. Real-world materials have rich surface texture, geometry, lighting conditions, and clutter, which combine to make the problem particularly difficult. We approach this problem by combining a new large-scale dataset with deep learning. MINC is an order of magnitude larger than previous material databases, while being more diverse and well-sampled across its 23 categories. Using MINC, we train CNNs for two tasks: classifying materials from patches, and simultaneous material recognition and segmentation in full images. For patch-based classification on MINC we found that the best performing CNN architectures can achieve 85.2% mean class accuracy. We convert these trained CNN classifiers into an efficient fully convolutional framework combined with a fully connected conditional random field (CRF) to predict the material at every pixel in an image, achieving 73.1% mean class accuracy. Our experiments demonstrate that having a large, well-sampled dataset such as MINC is crucial for real-world material recognition and segmentation. MINC is used today by academic and industry groups, is one of the largest open deep learning datasets,

and our material recognition CNN is best in class.

Chapter 3 presents a novel method for assigning annotations to images by the agreement of multiple consensuses of small cliques of crowdsource workers. A key step in deep learning is acquiring accurate training labels from humans. Collecting large-scale datasets is challenging, and the learning algorithms are only as good as the data they train on. Training labels are often obtained by taking the majority label from independent crowdsourced workers using platforms such as Amazon Mechanical Turk. However, the accuracy of the resulting labels can vary, with the hardest-to-label samples having prohibitively low accuracy. In cases where independent worker annotations are poor more accurate labels can be obtained by having workers collaborate. We describe an approach that reduces error by 37.8% on two different datasets at a cost of $0.10 or $0.17 per label. The higher cost is justified because our method does not need to be run on the entire dataset. Ultimately, our method enables us to more accurately annotate images and build more challenging training datasets for learning algorithms.

Chapter 4 presents Deep Feature Interpolation (DFI), a method for repurposing a trained CNN to photorealistically change the attributes of a face or fill in the missing parts of an image. Training a deep CNN is difficult. Large amounts of accurate labeled data and specialized knowledge is needed to train a high-performance CNN. An exciting direction in scene understanding is repurposing existing CNNs to new tasks without any additional training. These training-free approaches can be surprisingly effective. DFI relies only on simple linear interpolation of deep convolutional features from a pretrained CNN. We show that despite its simplicity, DFI can perform high-level semantic transformations like

"make older/younger", "make bespectacled", "add smile", among others, surprisingly well—sometimes even matching or outperforming the state-of-the-art. This is particularly unexpected as DFI requires no specialized network architecture or even any deep network to be trained for these tasks. DFI therefore can be used as a new baseline to evaluate more complex algorithms and provides a practical answer to the question of which image transformation tasks are still challenging after the advent of deep learning. DFI has added to our knowledge of scene understanding CNNs while also making practical contributions to the fast-growing area of automated high-resolution photorealistic image manipulation.

These works explore two facets of scene understanding: material segmentation and photorealistic image editing. They also benefit practitioners of deep learning by describing new ways to gather large-scale crowdsourced training data and presenting surprising properties of deep feature activations. These works also enable practical applications for identifying materials and modifying portraits.

CHAPTER 2

MATERIAL RECOGNITION IN THE WILD

2.1 Introduction

Material recognition plays a critical role in our understanding of and interactions with the world. To tell whether a surface is easy to walk on, or what kind of grip to use to pick up an object, we must recognize the materials that make up our surroundings. Automatic material recognition can be useful in a variety of applications, including robotics, product search, and image editing for interior design. But recognizing materials in real-world images is very challenging. Many categories of materials, such as fabric or wood, are visually very rich and span a diverse range of appearances. Materials can further vary in appearance due to lighting and shape. Some categories, such as plastic and ceramic, are often smooth and featureless, requiring reasoning about subtle cues or context to differentiate between them.

Large-scale datasets (e.g., ImageNet [108], SUN [146, 100] and Places [157]) combined with convolutional neural networks (CNNs) have been key to recent breakthroughs in object recognition and scene classification. Material recognition is similarly poised for advancement through large-scale data and learning. To date, progress in material recognition has been facilitated by moderate-sized datasets like the Flickr Material Database (FMD) [118]. FMD contains ten material categories, each with 100 samples drawn from Flickr photos. These images were carefully selected to illustrate a wide range of appearances for these categories. FMD has been used in research on new features and learning methods for material perception and recognition [83, 51, 101, 117]. While FMD was an im-

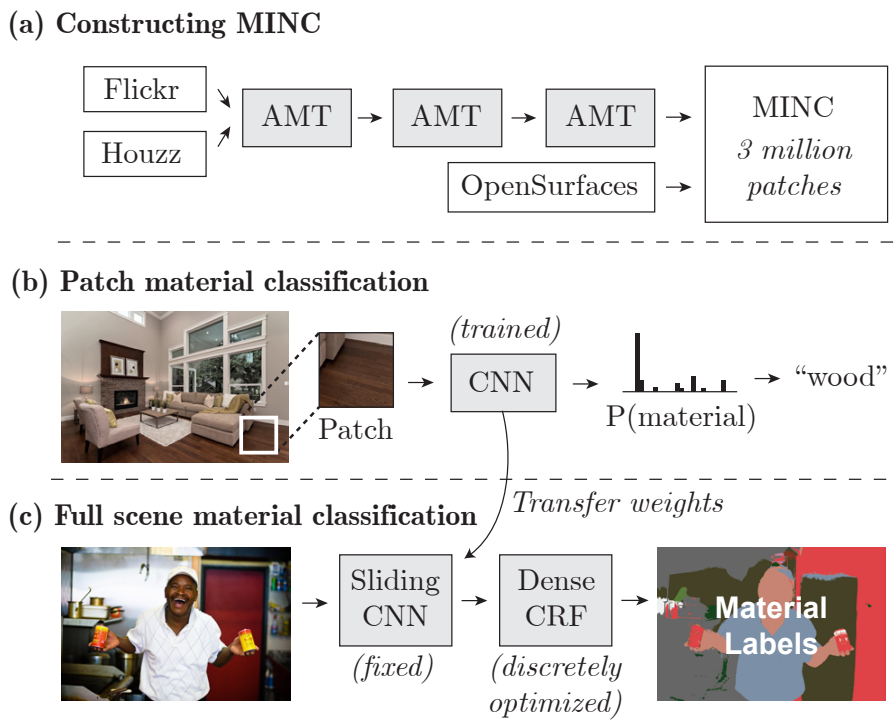


Figure 2.1: **Overview. (a)** We construct a new dataset by combining OpenSurfaces [5] with a novel three-stage Amazon Mechanical Turk (AMT) pipeline. **(b)** We train various CNNs on patches from MINC to predict material labels. **(c)** We transfer the weights to a fully convolutional CNN to efficiently generate a probability map across the image; we then use a fully connected CRF to predict the material at every pixel.

portant step towards material recognition, it is not sufficient for classifying materials in real-world imagery. This is due to the relatively small set of categories, the relatively small number of images per category, and also because the dataset has been designed around hand-picked iconic images of materials. The OpenSurfaces dataset [5] addresses some of these problems by introducing 105,000 material segmentations from real-world images, and is significantly larger than FMD. However, in OpenSurfaces many material categories are under-sampled, with only tens of images.

A major contribution of this work is a new, well-sampled material dataset, called the **Materials in Context Database** (MINC), with **3 million** material sam-

ples. MINC is more diverse, has more examples in less common categories, and is much larger than existing datasets. MINC draws data from both Flickr images, which include many "regular" scenes, as well as Houzz images from professional photographers of staged interiors. These sources of images each have different characteristics that together increase the range of materials that can be recognized. See Figure 2.2 for examples of our data. We make our full dataset available online at http://minc.cs.cornell.edu/.

We use this data for material recognition by training different CNN architectures on this new dataset. We perform experiments that illustrate the effect of network architecture, image context, and training data size on subregions (i.e., patches) of a full scene image. Further, we build on our patch classification results and demonstrate simultaneous material recognition and segmentation of an image by performing dense classification over the image with a fully connected conditional random field (CRF) model [68]. By replacing the fully connected layers of the CNN with convolutional layers [114], the computational burden is significantly lower than a naive sliding window approach.

In summary, we make two new contributions:

- We introduce a new material dataset, MINC, and 3-stage crowdsourcing pipeline for efficiently collecting millions of click labels (Section 2.3.2).
- Our new semantic segmentation method combines a fully-connected CRF with unary predictions based on CNN learned features (Section 2.4.2) for simultaneous material recognition and segmentation.

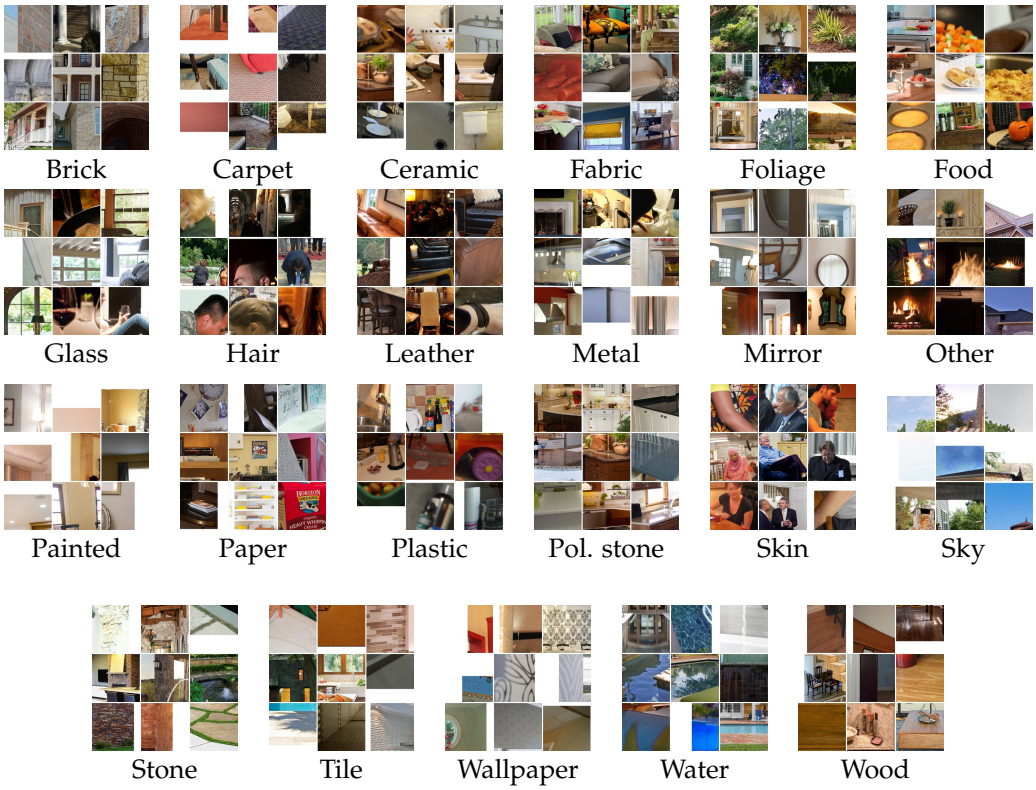


Figure 2.2: Example patches from all 23 categories of the Materials in Context Database (MINC). Note that we sample patches so that the patch center is the material in question (and not necessarily the entire patch). See Table 2.1 for the size of each category.

2.2 Prior Work

Material Databases. Much of the early work on material recognition focused on classifying specific instances of textures or material samples. For instance, the CUReT [18] database contains 61 material samples, each captured under 205 different lighting and viewing conditions. This led to research on the task of instance-level texture or material classification [77, 133], and an appreciation of the challenges of building features that are invariant to pose and illumination. Later, databases with more diverse examples from each material category began to appear, such as KTH-TIPS [46, 10], and led to explorations of how to gener-

alize from one example of a material to another—from one sample of wood to a completely different sample, for instance. Real-world texture attributes have also recently been explored [16].

In the domain of categorical material databases, Sharan et al. released FMD [118] (described above). Subsequently, Bell et al. released OpenSurfaces [5] which contains over 20,000 real-world scenes labeled with both materials and objects, using a multi-stage crowdsourcing pipeline. Because OpenSurfaces images are drawn from consumer photos on Flickr, material samples have real-world context, in contrast to prior databases (CUReT, KTH-TIPS, FMD) which feature cropped stand-alone samples. While OpenSurfaces is a good starting point for a material database, we substantially expand it with millions of new labels.

Material recognition. Much prior work on material recognition has focused on the classification problem (categorizing an image patch into a set of material categories), often using hand-designed image features. For FMD, Liu et al. [83] introduced reflectance-based edge features in conjunction with other general image features. Hu et al. [51] proposed features based on variances of oriented gradients. Qi et al. [101] introduced a pairwise local binary pattern (LBP) feature. Li et al. [80] synthesized a dataset based on KTH-TIPS2 and built a classifier from LBP and dense SIFT. Timofte et al. [131] proposed a classification framework with minimal parameter optimization. Schwartz and Nishino [113] introduced material traits that incorporate learned convolutional auto-encoder features. Recently, Cimpoi et al. [16] developed a CNN and improved Fisher vector (IFV) classifier that achieves state-of-the-art results on FMD and KTH-TIPS2. Finally, it has been shown that jointly predicting objects and materials

can improve performance [51, 155].

Convolutional neural networks. While CNNs have been around for a few decades, with early successes such as LeNet [75], they have only recently led to state-of-the-art results in object classification and detection, leading to enormous progress. Driven by the ILSVRC challenge [108], we have seen many successful CNN architectures [151, 114, 128, 120], led by the work of Krizhevsky et al. on their SuperVision (aka AlexNet) network [70], with more recent architectures including GoogLeNet [128]. In addition to image classification, CNNs are the state-of-the-art for detection and localization of objects, with recent work including R-CNNs [41], OverFeat [114], and VGG [120]. Finally, relevant to our goal of per-pixel material segmentation, Farabet et al. [33] use a multi-scale CNN to predict the class at every pixel in a segmentation. Oquab et al. [97] employ a sliding window approach to localize patch classification of objects. We build on this body of work in deep learning to solve our problem of material recognition and segmentation.

2.3 The Materials in Context Database (MINC)

We now describe the methodology that went into building our new material database. Why a new database? We needed a dataset with the following properties:

- **Size**: It should be sufficiently large that learning methods can generalize beyond the training set.

- **Well-sampled**: Rare categories should be represented with a large number of examples.
- **Diversity**: Images should span a wide range of appearances of each material in real-world settings.
- **Number of categories**: It should contain many different materials found in the real world.

2.3.1 Sources of data

We decided to start with the public, crowdsourced OpenSurfaces dataset [5] as the seed for MINC since it is drawn from Flickr imagery of everyday, real-world scenes with reasonable diversity. Furthermore, it has a large number of categories and the most samples of all prior databases.

While OpenSurfaces data is a good start, it has a few limitations. Many categories in OpenSurfaces are not well sampled. While the largest category, *wood*, has nearly 20K samples, smaller categories, such as *water*, have only tens of examples. This imbalance is due to the way the OpenSurfaces dataset was annotated; workers on Amazon Mechanical Turk (AMT) were free to choose any material subregion to segment. Workers often gravitated towards certain common types of materials or salient objects, rather than being encouraged to label a diverse set of materials. Further, the images come from a single source (Flickr).

We decided to augment OpenSurfaces with substantially more data, especially for underrepresented material categories, with the initial goal of gathering at least 10K samples per material category. We decided to gather this data from

another source of imagery, professional photos on the interior design website Houzz. Our motivation for using this different source of data was that, despite Houzz photos being more "staged" (relative to Flickr photos), they actually represent a larger variety of materials. For instance, Houzz photos contain a wide range of types of polished stone. With these sources of image data, we now describe how we gather material annotations.

2.3.2 Segments, Clicks, and Patches

What specific kinds of material annotations make for a good database? How should we collect these annotations? The type of annotations to collect is guided in large part by the tasks we wish to generate training data for. For some tasks such as scene recognition, whole-image labels can suffice [146, 157]. For object detection, labeled bounding boxes as in PASCAL are often used [31]. For segmentation or scene parsing tasks, per-pixel segmentations are required [109, 45]. Each style of annotation comes with a cost proportional to its complexity. For materials, we decided to focus on two problems, guided by prior work:

- **Patch material classification.** Given an image patch, what kind of material is it at the center?
- **Full scene material classification.** Given a full image, produce a full per-pixel segmentation and labeling. Also known as *semantic segmentation* or *scene parsing* (but in our case, focused on materials). Note that classification can be a component of segmentation, e.g., with sliding window approaches.

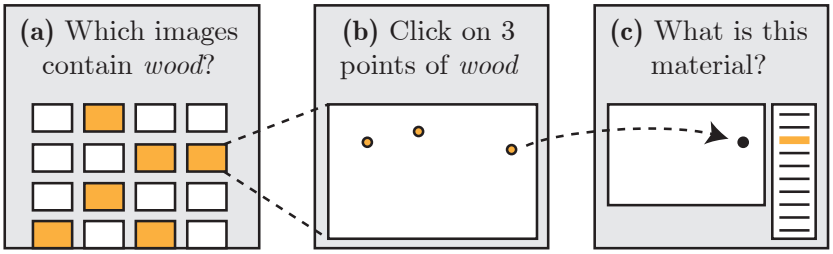


Figure 2.3: **AMT pipeline schematic for collecting clicks. (a)** Workers filter by images that contain a certain material, **(b)** workers click on materials, and **(c)** workers validate click locations by re-labeling each point. Example responses are shown in orange.

Segments. OpenSurfaces contains material segmentations—carefully drawn polygons that enclose same-material regions. To form the basis of MINC, we selected OpenSurfaces segments with high confidence (inter-worker agreement) and manually curated segments with low confidence, giving a total of 72K shapes. To better balance the categories, we manually segmented a few hundred extra samples for *sky*, *foliage* and *water*.

Since some of the OpenSurfaces categories are difficult for humans, we consolidated these categories. We found that many AMT workers could not disambiguate *stone* from *concrete*, *clear plastic* from *opaque plastic*, and *granite* from *marble*. Therefore, we merged these into *stone*, *plastic*, and *polished stone* respectively. Without this merging, many ground truth examples in these categories would be incorrect. The final list of 23 categories is shown in Table 2.1. The category *other* is different in that it was created by combining various smaller categories.

Clicks. Since we want to expand our dataset to millions of samples, we decided to augment OpenSurfaces segments by collecting *clicks*: single points in an image along with a material label, which are much cheaper and faster to collect. Figure 2.3 shows our pipeline for collecting clicks.

Initially, we tried asking workers to click on examples of a given material in a photo. However, we found that workers would get frustrated if the material was absent in too many of the photos. Thus, we added an initial first stage where workers filter out such photos. To increase the accuracy of our labels, we verify the click labels by asking different workers to specify the material for each click without providing them with the label from the previous stage.

To ensure that we obtain high quality annotations and avoid collecting labels from workers who are not making an effort, we include secret known answers (sentinels) in the first and third stages, and block workers with an accuracy below 50% and 85% respectively. We do not use sentinels in the second stage since it would require per-pixel ground truth labels, and it turned out not to be necessary. Workers generally performed all three tasks so we could identify bad workers in the first or third task.

Material clicks were collected for both OpenSurfaces images and the new Houzz images. This allowed us to use labels from OpenSurfaces to generate the sentinel data; we included 4 sentinels per task. With this streamlined pipeline we collected 2,341,473 annotations at an average cost of $0.00306 per annotation (stage 1: $0.02 / 40 images, stage 2: $0.10 / 50 images, 2, stage 3: $0.10 / 50 points).

Patches. Labeled segments and clicks form the core of MINC. For training CNNs and other types of classifiers, it is useful to have data in the form of fixed-sized patches. We convert both forms of data into a unified dataset format: square image patches. We use a *patch center* and *patch scale* (a multiplier of the smaller image dimension) to define the image subregion that makes a patch. For our patch classification experiments, we use 23.3% of the smaller image

Patches	Category	Patches	Category	Patches	Category
564,891	Wood	114,085	Polished stone	35,246	Skin
465,076	Painted	98,891	Carpet	29,616	Stone
397,982	Fabric	83,644	Leather	28,108	Ceramic
216,368	Glass	75,084	Mirror	26,103	Hair
188,491	Metal	64,454	Brick	25,498	Food
147,346	Tile	55,364	Water	23,779	Paper
142,150	Sky	39,612	Other	14,954	Wallpaper
120,957	Foliage	38,975	Plastic		

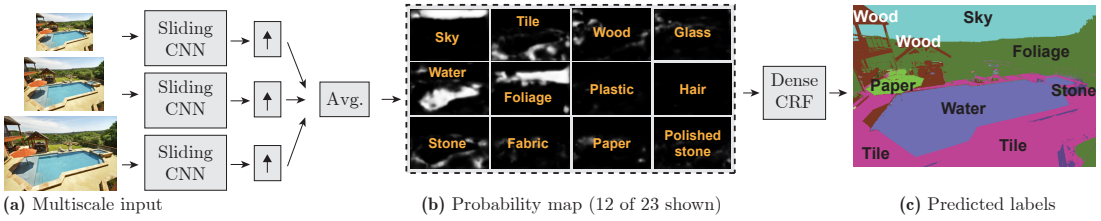
Table 2.1: **MINC patch counts by category.** Patches were created from both OpenSurfaces segments and our newly collected *clicks*. See Section 2.3.2 for details.

dimension. Increasing the patch scale provides more context but reduces the spatial resolution. Later in Section 2.5 we justify our choice with experiments that vary the patch scale for AlexNet.

We place a patch centered around each click label. For each segment, if we were to place a patch at every interior pixel then we would have a very large and redundant dataset. Therefore, we Poisson-disk subsample each segment, separating patch centers by at least 9.1% of the smaller image dimension. These segments generated 655,201 patches (an average of 9.05 patches per segment). In total, we generated 2,996,674 labeled patches from 436,749 images. Patch counts are shown in Table 2.1, and example patches from various categories are illustrated in Figure 2.2.

2.4 Material recognition in real-world images

Our goal is to train a system that recognizes the material at every pixel in an image. We split our training procedure into multiple stages and analyze the



(a) Multiscale input **(b)** Probability map (12 of 23 shown) **(c)** Predicted labels

Figure 2.4: **Pipeline for full scene material classification.** An image **(a)** is resized to multiple scales $[1/\sqrt{2}, 1, \sqrt{2}]$. The same sliding CNN predicts a probability map **(b)** across the image for each scale; the results are upsampled and averaged. A fully connected CRF predicts a final label for each pixel **(c)**. This example shows predictions from a single GoogLeNet converted into a sliding CNN (no average pooling).

performance of the network at each stage. First, we train a CNN that produces a single prediction for a given input patch. Then, we convert the CNN into a sliding window and predict materials on a dense grid across the image. We do this at multiple scales and average to obtain a unary term. Finally, a dense CRF [68] combines the unary term with fully connected pairwise reasoning to output per-pixel material predictions. The entire system is depicted in Figure 2.1, and described more below.

2.4.1 Training procedure

MINC contains 3 million patches that we split into training, validation and test sets. Randomly splitting would result in nearly identical patches (e.g., from the same OpenSurfaces segment) being put in training and test, thus inflating the test score. To prevent correlation, we group photos into clusters of near-duplicates, then assign each cluster to one of train, validate or test. We make sure that there are at least 75 segments of each category in the test set to ensure there are enough segments to evaluate segmentation accuracy. To detect near-duplicates, we compare AlexNet CNN features computed from each photo (see

the supplemental for details). For exact duplicates, we discard all but one of the copies.

We train all of our CNNs by fine-tuning the network starting from the weights obtained by training on 1.2 million images from ImageNet (ILSVRC 2012). When training AlexNet, we use stochastic gradient descent with batch-size 128, dropout rate 0.5, momentum 0.9, and a base learning rate of 10^{-3} that decreases by a factor of 0.25 every 50,000 iterations. For GoogLeNet, we use batchsize 69, dropout 0.4, and learning rate $\alpha_t = 10^{-4}\sqrt{1 - t/250000}$ for iteration t.

Our training set has a different number of examples per class, so we cycle through the classes and randomly sample an example from each class. Failing to properly balance the examples results in a 5.7% drop in mean class accuracy (on the validation set). Further, since it has been shown to reduce overfitting, we randomly augment samples by taking crops (227×227 out of 256×256), horizontal mirror flips, spatial scales in the range $[1/\sqrt{2}, \sqrt{2}]$, aspect ratios from 3:4 to 4:3, and amplitude shifts in $[0.95, 1.05]$. Since we are looking at local regions, we subtract a per-channel mean (R: 124, G: 117, B: 104) rather than a mean image [70].

2.4.2 Full scene material classification

Figure 2.4 shows an overview of our method for simultaneously segmenting and recognizing materials. Given a CNN that can classify individual points in the image, we convert it to a sliding window detector and densely classify a grid across the image. Specifically, we replace the last fully connected layers

with convolutional layers, so that the network is fully convolutional and can classify images of any shape. After conversion, the weights are fixed and not fine-tuned. With our converted network, the strides of each layer cause the network to output a prediction every 32 pixels. We obtain predictions every 16 pixels by shifting the input image by half-strides (16 pixels). While this appears to require 4x the computation, Sermanet et al. [114] showed that the convolutions can be reused and only the pool5 through fc8 layers need to be recomputed for the half-stride shifts. Adding half-strides resulted in a minor 0.2% improvement in mean class accuracy across segments (after applying the dense CRF, described below), and about the same mean class accuracy at click locations.

The input image is resized so that a patch maps to a 256x256 square. Thus, for a network trained at patch scale s, the resized input has smaller dimension $d = 256/s$. Note that d is inversely proportional to scale, so increased context leads to lower spatial resolution. We then add padding so that the output probability map is aligned with the input when upsampled. We repeat this at 3 different scales (smaller dimension $d/\sqrt{2}$, d, $d\sqrt{2}$), upsample each output probability map with bilinear interpolation, and average the predictions. To make the next step more efficient, we upsample the output to a fixed smaller dimension of 550.

We then use the dense CRF of Krähenbühl et al. [68] to predict a label at every pixel, using the following energy:

$$E(x \mid \mathbf{I}) = \sum_i \psi_i(x_i) + \sum_{i<j} \psi_{ij}(x_i, x_j) \tag{2.1}$$

$$\psi_i(x_i) = -\log p_i(x_i) \tag{2.2}$$

$$\psi_{ij}(x_i, x_j) = w_p \, \delta(x_i \neq x_j) \, k(\mathbf{f}_i - \mathbf{f}_j) \tag{2.3}$$

Architecture	Validation	Test
AlexNet [70]	82.2%	81.4%
GoogLeNet [128]	**85.9%**	**85.2%**
VGG-16 [120]	85.6%	84.8%

Table 2.2: **Patch material classification results.** Mean class accuracy for different CNNs trained on MINC. See Section 2.5.1.

Sky	97.3%	Food	90.4%	Wallpaper	83.4%	Glass	78.5%
Hair	95.3%	Leather	88.2%	Tile	82.7%	Fabric	77.8%
Foliage	95.1%	Other	87.9%	Ceramic	82.7%	Metal	77.7%
Skin	93.9%	Pol. stone	85.8%	Stone	82.7%	Mirror	72.0%
Water	93.6%	Brick	85.1%	Paper	81.8%	Plastic	70.9%
Carpet	91.6%	Painted	84.2%	Wood	81.3%		

Table 2.3: **Patch test accuracy by category.** CNN: GoogLeNet. See the supplemental material for a full confusion matrix.

where ψ_i is the unary energy (negative log of the aggregated softmax probabilities) and ψ_{ij} is the pairwise term that connects every pair of pixels in the image. We use a single pairwise term with a Potts label compatibility term δ weighted by w_p and unit Gaussian kernel k. For the features $\mathbf{f}_i$, we convert the RGB image to $L^*a^*b^*$ and use color (I_i^L, I_i^a, I_i^b) and position (p^x, p^y) as pairwise features for each pixel:

$$\mathbf{f}_i = \left[\frac{p_i^x}{\theta_p\, d}, \frac{p_i^y}{\theta_p\, d}, \frac{I_i^L}{\theta_L}, \frac{I_i^a}{\theta_{ab}}, \frac{I_i^b}{\theta_{ab}} \right], \tag{2.4}$$

where d is the smaller image dimension. Figure 2.4 shows an example unary term p_i and the resulting segmentation x.

2.5 Experiments and Results

2.5.1 Patch material classification

In this section, we evaluate the effect of many different design decisions for training methods for material classification and segmentation, including various CNN architectures, patch sizes, and amounts of data.

CNN Architectures. Our ultimate goal is full material segmentation, but we are also interested in exploring which CNN architectures give the best results for classifying single patches. Among the networks and parameter variations we tried we found the best performing networks were AlexNet [70], VGG-16 [120] and GoogLeNet [128]. AlexNet and GoogLeNet are re-implementations by BVLC [56], and VGG-16 is configuration D (a 16 layer network) of [120]. All models were obtained from the Caffe Model Zoo [56]. Our experiments use AlexNet for evaluating material classification design decisions and combinations of AlexNet and GoogLeNet for evaluating material segmentation. Tables 2.2 and 2.3 summarize patch material classification results on our dataset. Figure 2.10 shows correct and incorrect predictions made with high confidence.

Input patch scale. To classify a point in an image we must decide how much context to include around it. The context, expressed as a fraction of image size, is the patch scale. A priori, it is not clear which scale is best since small patches have better spatial resolution, but large patches have more contextual information. Holding patch centers fixed we varied scale and evaluated classification accuracy with AlexNet. Results and a visualization of patch scales are shown in Figure 2.5. Scale 32% performs the best. Individual categories had peaks at mid-

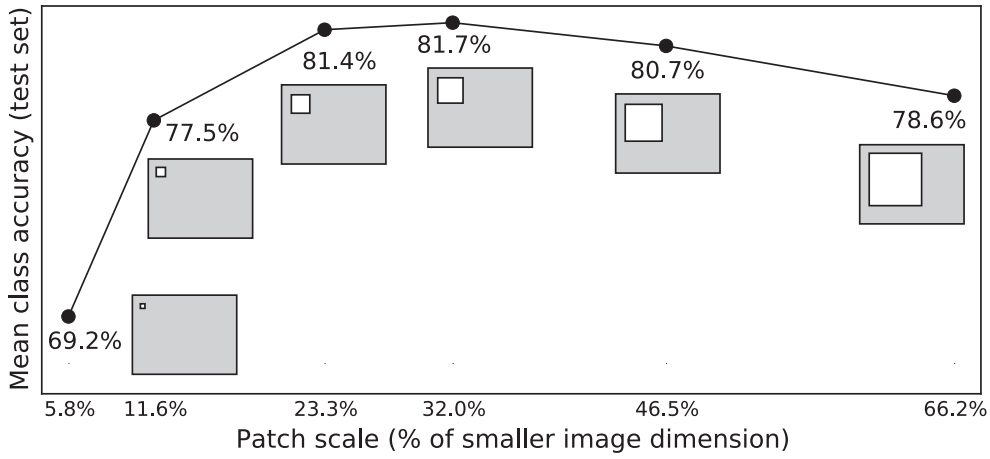


Figure 2.5: **Varying patch scale**. We train/test patches of different scales (the patch locations do not vary). The optimum is a trade-off between context and spatial resolution. CNN: AlexNet.

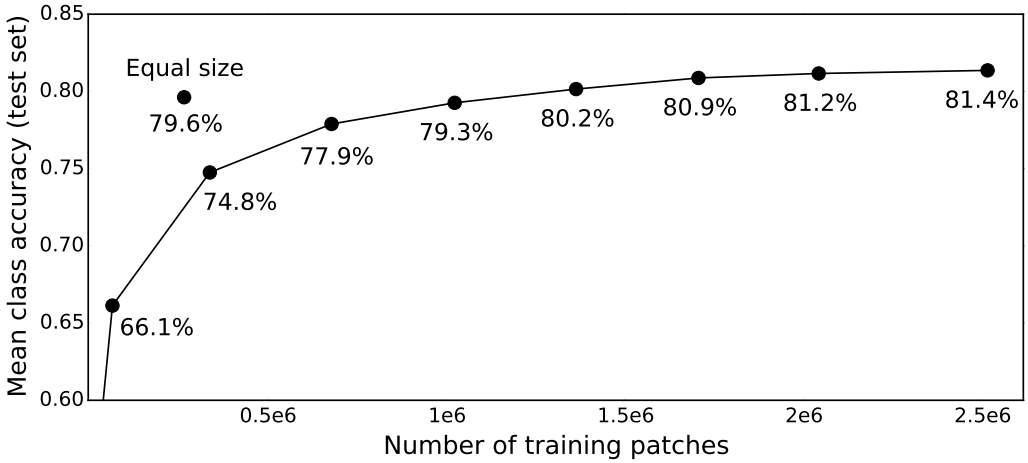


Figure 2.6: **Varying database size.** Patch accuracy when trained on random subsets of MINC. *Equal size* is using equal samples per category (size determined by smallest category). CNN: AlexNet.

dle scale with some exceptions; we find that *mirror*, *wallpaper* and *sky* improve with increasing context (Figure 2.7). We used 23.3% (which has nearly the same accuracy but higher spatial resolution) for our experiments.

Dataset size. To measure the effect of size on patch classification accuracy we trained AlexNet with patches from randomly sampled subsets of all 369,104

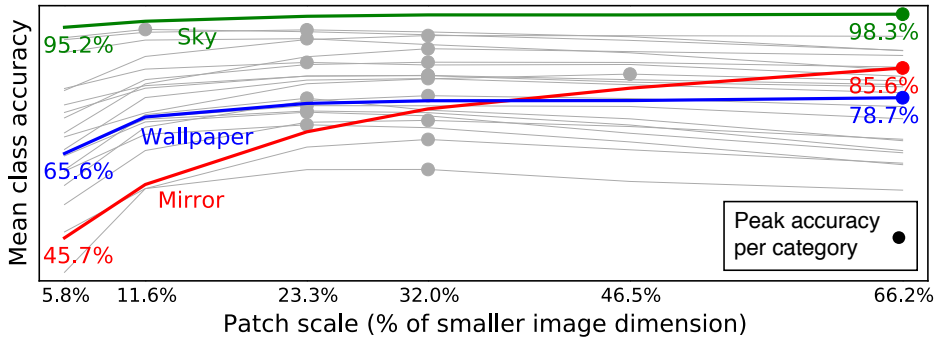
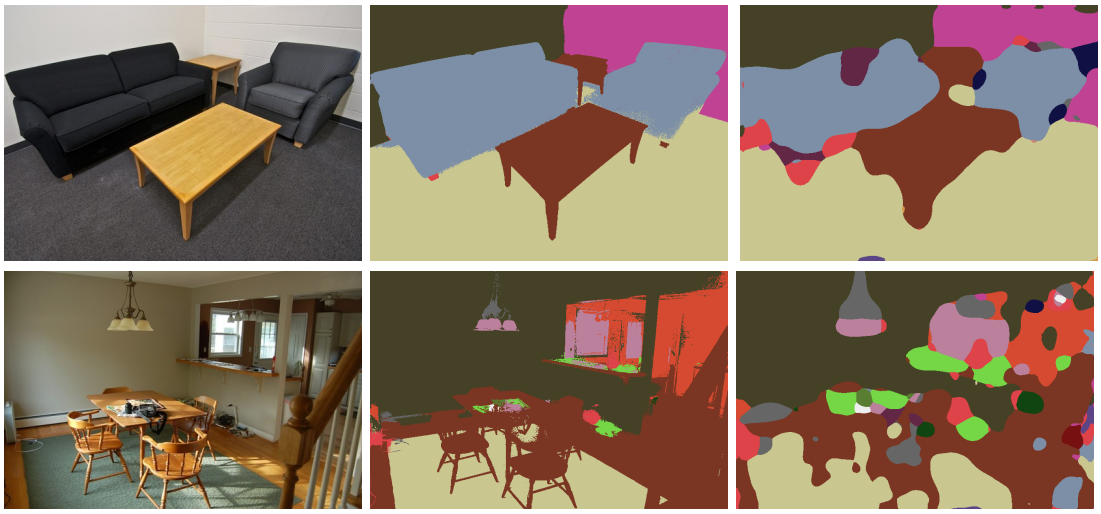


Figure 2.7: **Accuracy vs patch scale by category**. Dots: peak accuracy for each category; colored lines: *sky*, *wallpaper*, *mirror*; gray lines: other categories. CNN: AlexNet. While most materials are optimally recognized at 23.3% or 32% patch scale, recognition of *sky*, *wallpaper* and *mirror* improve with increasing context.

Figure 2.8: **Full-scene material classification examples:** high-accuracy test set predictions by our method. CNN: GoogLeNet (with the average pooling layer removed). Right: legend for material colors. See Table 2.4 for quantitative evaluation.

training images and tested on our full test set (Figure 2.6). As expected, using more data improved performance. In addition, we still have not saturated performance with 2.5 million training patches; even higher accuracies may be possible with more training data (though with diminishing returns).

Dataset balance. Although we've shown that more data is better we also find that a balanced dataset is more effective. We trained AlexNet with all patches of our smallest category (*wallpaper*) and randomly sampled the larger categories (the largest, *wood*, being 40x larger) to be equal size. We then measured mean class accuracy on the same full test set. As shown in Figure 2.6, "Equal size" is



Input image (test set) | **(a)** Labels from CRF trained on segments | **(b)** Labels from CRF trained on clicks

Figure 2.9: **Optimizing for click accuracy leads to sloppy boundaries.** In **(a)**, we optimize for mean class accuracy across segments, resulting in high quality boundaries. In **(b)**, we optimize for mean class accuracy at click locations. Since the clicks are not necessarily close to object boundaries, there is no penalty for sloppy boundaries. CNN: GoogLeNet (without average pooling).

more accurate than a dataset of the same size and just 1.7% lower than the full training set (which is 9x larger). This result further demonstrates the value of building up datasets in a balanced manner, focusing on expanding the smallest, least common categories.

2.5.2 Full scene material segmentation

The full test set for our patch dataset contains 41,801 photos, but most of them contain only a few labels. Since we want to evaluate the per-pixel classification performance, we select a subset of 5,000 test photos such that each photo contains a large number of segments and clicks, and small categories are well sampled. We greedily solve for the best such set of photos. We similarly select 2,500 of 25,844 validation photos. Our splits for all experiments are included on-

Architecture		**(a)** Segments only		**(b)** Clicks only	
		Class	Total	Class	Total
AlexNet	Scale: 11.6%	64.3%	72.6%	79.9%	77.2%
AlexNet	Scale: 23.3%	69.6%	76.6%	**83.3%**	**81.1%**
AlexNet	Scale: 32.0%	**70.1%**	**77.1%**	83.2%	80.7%
AlexNet	Scale: 46.5%	69.6%	75.4%	80.8%	77.7%
AlexNet	Scale: 66.2%	67.7%	72.0%	77.2%	72.6%
GoogLeNet	7x7 avg. pool	64.4%	71.6%	63.6%	63.4%
GoogLeNet	5x5 avg. pool	67.6%	74.6%	70.9%	69.8%
GoogLeNet	3x3 avg. pool	**70.4%**	77.7%	76.1%	74.7%
GoogLeNet	No avg. pool	**70.4%**	**78.8%**	**79.1%**	**77.4%**
Ensemble	2 CNNs	**73.1%**	**79.8%**	84.5%	83.1%
Ensemble	3 CNNs	**73.1%**	79.3%	**85.9%**	**83.5%**
Ensemble	4 CNNs	72.1%	78.4%	85.8%	83.2%
Ensemble	5 CNNs	71.7%	78.3%	85.5%	83.2%

Table 2.4: **Full scene material classification results.** Mean class and total accuracy on the test set. When training, we optimize the CRF parameters for mean class accuracy, but report both mean class and total accuracy (mean accuracy across all examples). In one experiment **(a)**, we train and test only on segments; in a separate experiment **(b)**, we train and test only on clicks. Accuracies for segments are averaged across all pixels that fall in that segment.

line with the dataset. To train the CRF for our model, we try various parameter settings (θ_p, θ_{ab}, θ_L, w_p) and select the model that performs best on the validation set. In total, we evaluate 1799 combinations of CNNs and CRF parameters. See the supplemental material for a detailed breakdown.

We evaluate multiple versions of GoogLeNet: both the original architecture and a version with the average pooling layer (at the end) changed to 5x5, 3x3, and 1x1 (i.e. no average pooling). We evaluate AlexNet trained at multiple patch scales (Figure 2.5). When using an AlexNet trained at a different scale, we keep the same scale for testing. We also experiment with ensembles of GoogLeNet and AlexNet, combined with either arithmetic or geometric mean.

Since we have two types of data, *clicks* and *segments*, we run two sets of ex-

periments: (a) we train and test only on segments, and in a separate experiment (b) we train and test only on clicks. These two training objectives result in very different behavior, as illustrated in Figure 2.9. In experiment (a), the accuracy across segments are optimized, producing clean boundaries. In experiment (b), the CRF maximizes accuracy only at click locations, thus resulting in sloppy boundaries. As shown in Table 2.4, the numerical scores for the two experiments are also very different: segments are more challenging than clicks. While clicks are sufficient to train a CNN, they are not sufficient to train a CRF.

Focusing on segmentation accuracy, we see from Table 2.4(a) that our best single model is GoogLeNet without average pooling (6% better than with pooling). The best ensemble is 2 CNNs: GoogLeNet (no average pooling) and AlexNet (patch scale: 46.5%), combined with arithmetic mean. Larger ensembles perform worse since we are averaging worse CNNs. In Figure 2.8, we show example labeling results on test images.

2.5.3 Comparing MINC to FMD

Compared to FMD, the size and diversity of MINC is valuable for classifying real-world imagery. Table 2.5 shows the effect of training on all of FMD and testing on MINC (and vice versa). The results suggests that training on FMD alone is not sufficient for real-world classification. Though it may seem that our dataset is "easy," since the best classification scores are lower for FMD than for MINC, we find that difficulty is in fact closely tied to dataset size (Section 2.5.1). Taking 100 random samples per category, AlexNet achieves 54.2 ± 0.7% on MINC (64.6 ± 1.3% when considering only the 10 FMD categories) and

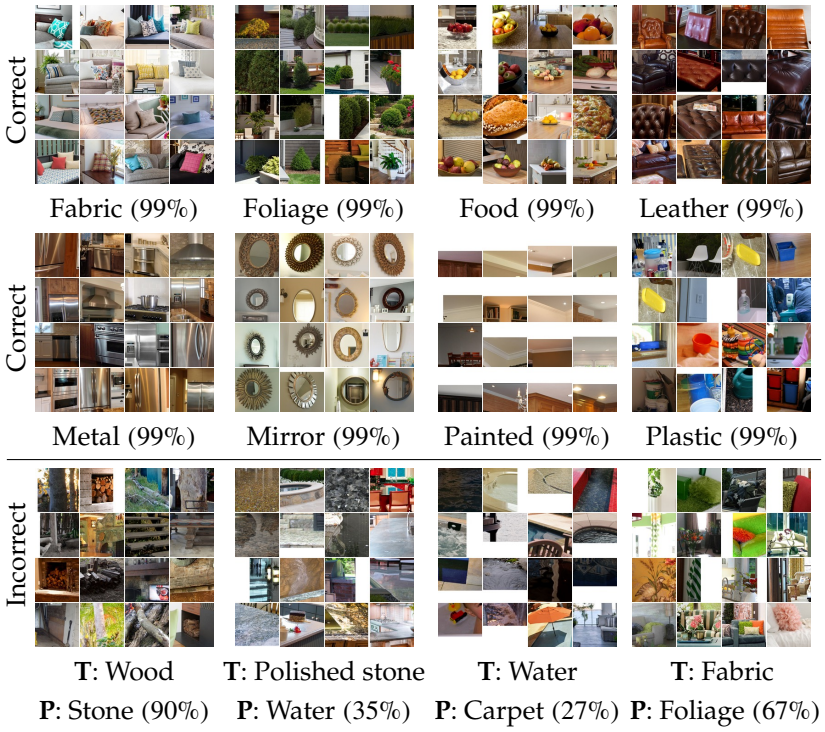


Figure 2.10: **High confidence predictions.** Top two rows: correct predictions. Bottom row: incorrect predictions (**T**: true, **P**: predicted). Percentages indicate confidence (the predictions shown are at least this confident). CNN: GoogLeNet.

66.5% on FMD.

2.5.4 Comparing CNNs with prior methods

Cimpoi [16] is the best prior material classification method on FMD. We find that by replacing DeCAF features [23] with oversampled AlexNet features we can improve on their FMD results. We then show that on MINC, a finetuned CNN is even better.

To improve on [16], we take their SIFT_IFV, combine it with AlexNet fc7 features, and add oversampling [70] (see supplemental for details). With a linear

		Test		
		FMD	MINC	*(10 categories*
Train	FMD	66.5%	26.1%	*in common)*
	MINC	41.7%	85.0%	

Table 2.5: **Cross-dataset experiments.** We train on one dataset and test on another dataset. Since MINC contains 23 categories, we limit MINC to the 10 categories in common. CNN: AlexNet.

Method	**Accuracy**	**Trials**
Sharan et al. [117]	57.1 ± 0.6%	14 splits
Cimpoi et al. [16]	67.1 ± 0.4%	14 splits
Fine-tuned AlexNet	66.5 ± 1.5%	5 folds
SIFT_IFV+fc7	**69.6 ± 0.3%**	10 splits

Table 2.6: **FMD experiments.** By replacing DeCAF features with oversampled AlexNet features we improve on the best FMD result.

SVM we achieve 69.6 ± 0.3% on FMD. Previous results are listed in Table 2.6.

Having found that SIFT_IFV+fc7 is the new best on FMD, we compare it to a finetuned CNN on a subset of MINC (2500 patches per category, one patch per photo). Fine-tuning AlexNet achieves 76.0 ± 0.2% whereas SIFT_IFV+fc7 achieves 67.4 ± 0.5% with a linear SVM (oversampling, 5 splits). This experiment shows that a finetuned CNN is a better method for MINC than SIFT_IFV+fc7.

2.6 Conclusion

Material recognition is a long-standing, challenging problem. We introduce a new large, open, material database, MINC, that includes a diverse range of materials of everyday scenes and staged designed interiors, and is at least an order of magnitude larger than prior databases. Using this large database we pro-

pose a method for simultaneous material classification and segmentation based on recent advances in deep learning, and achieve results that surpass prior attempts at material recognition.

Some lessons we have learned are:

- Training on a dataset which includes the surrounding context is crucial for real-world material classification.
- Labeled clicks are cheap and sufficient to train a CNN alone. However, to obtain high quality segmentation results, training a CRF on polygons results in much better boundaries than training on clicks.

Many future avenues of work remain. Expanding the dataset to a broader range of categories will require new ways to mine images that have more variety, and new annotation tasks that are cost-effective. Inspired by attributes for textures [16], in the future we would like to identify material attributes and expand our database to include them. We also believe that further exploration of joint material and object classification and segmentation will be fruitful [51] and lead to improvements in both tasks.

2.7 Impact

Our work in this chapter was published at CVPR 2015 [6]. In our work we proposed that collecting clicks from crowdsource workers was a more efficient way to gather training labels for semantic segmentation models. A work by Bearman et al. [4] found clicks to be more efficient than segmentations, strokes (aka

squiggles) and image-level labels. Our database of material annotations has been used to improve the prediction of haptic physical properties [35], acoustic physical properties [112] and propose material assignments for indoor digital scenes [12]. Our material recognition model parameters have been used to learn semantic segmentation of materials in 4D light-fields [136]. Our combination of a CNN with a dense CRF has inspired new models which more tightly combine the strengths of CNN prediction with edge refinement: state-of-the-art object segmentation [13], a recurrent neural network formulation of a CRF [156] and edge refinement with bilateral filtering [34].

Our database, trained models, and all experimental results are available online at http://minc.cs.cornell.edu/.

CHAPTER 3

CONSENSUS AGREEMENT GAMES

3.1 Introduction

Creating large-scale image datasets has proved crucial to enabling breakthrough performance on computer vision tasks [70]. A significant barrier to the creation of such datasets has been the human labor required to accurately annotate large collections of images. Increasingly such datasets have been labeled through innovations in the area of crowdsourcing and human computation, whereby the efforts of large numbers of often inexpert Internet-based workers are used to yield data of surprising accuracy [21, 59, 103, 109, 122, 134, 143]. The introduction of the Amazon Mechanical Turk crowdsourcing platform in 2006 in particular quickly led to its adoption in various image recognition tasks [3, 27, 122, 123, 21].

The most common approach seeks labels for each item from multiple workers and assigns as the item's label the "majority vote" among those provided by the workers [119, 121, 122]. Even as increasingly sophisticated approaches have been developed for aggregating the labels of independent workers there can still be significant variability in the quality of such data. Many samples receive very low agreement when labeled by multiple MTurk workers. For example, [5] collected approximately five labels per sample, and for those with 60% agreement (3 out of 5 agreement, after removing votes from low-quality workers) many of these *low-agreement samples* were mislabeled, making them unsuitable for training a high-accuracy model. As a result, [6] only used samples with at least 80% agreement (*high-agreement samples*).

Relying solely on high-agreement data can bias the data to easy-to-classify cases, which may not mirror the diversity of cases to which the trained model will be applied, negatively impacting the quality of the learned model. Further, low-agreement samples can represent cases that fall near decision boundaries, reflecting data that can be particularly valuable for improving model accuracy [79].

Ultimately, the problem is that MTurk workers have a high error rate on low-agreement data. If the *MTurk error rate* is the fraction of mislabed samples (compared to, for example, expert labelers or some other appropriate notion of ground truth), our goal is to reduce it so that low-agreement samples become more accurate, and thereby more useful for training computer vision models.

Reducing the MTurk error rate is not easy. The key characteristic of low-agreement data is that *independent* workers cannot agree on the label. Getting more answers from independent workers or encouraging them with agreement incentives does not get us better answers (as shown in the Experiments section). Instead, we take an approach where labels are assigned through a collaborative process involving multiple workers. We find our inspiration in two previous works. First, the graph consensus-finding work of [57, 61] showed that a network of workers can collectively reach consensus even when interactions are highly constrained. Next, the ESP Game [134] showed how to obtain labels from non-communicative workers by seeking agreement around a shared image. In this chapter, we show how to label images by casting it as a graph consensus problem which seeks agreement between multiple, independent consensus-finding cliques of workers. We find this pattern to be effective on difficult-to-label images.

Our approach achieves greater accuracy at a greater cost than majority voting, and thus the approach is intended for use in the context of creating large-scale databases of labeled images that are not biased towards easy-to-classify samples. This approach should be used *after* using majority voting to gather labels, *only* to refine the labels of low-agreement samples.

3.2 Related Work

Others have considered different ways in which the confidence in an item's annotation may differ across items, and its implications. For example, Galaxy Zoo created "clean" and "superclean" datasets by constraining data to those in which at least 80% or 95% of workers agree on an item's label [82], and [50] use worker disagreement so as to remove ambiguous data and improve the classification of sentiment in political blog excerpts. [20, 21, 98] consider how disagreement among obtained labels can be used to signal that more labels should be obtained for such items and [55] uses the uncertainty of a trained model on a dataset to target items for which additional labels can improve learning. [135] show that learning can be improved if workers are allowed to specify their own low confidence in labeling an item so that it can be routed to more expert workers, while [115] proposes a payment mechanism to incentivize workers to only respond to items they have confidence about. In settings where workers provide labels for multiple items it is possible to learn models of worker performance based on factors that include the extent of ambiguity of an item's annotation [2, 87, 138]. This work is complementary to such efforts, in that rather than persist with methods in which workers assign labels in isolation, we seek to improve annotation accuracy by employing consensus agreement games in which

workers act collaboratively to assign labels to items.

This work seeks more accurate annotations by engaging workers in a gaming setting similar to the ESP Game [134]. The ESP Game gives images to pairs of players then annotates the image and rewards the players if they enter identical tags. A number of variants to the ESP Game have also been proposed. The Make a Difference game [130] is similar to the ESP Game, but requires three workers to agree on a tag for it to be accepted. Further generalization to number of players and thresholds was made by [81]. KissKissBan [48] is a three-person game in which two players attempt to enter matching tags after a third player enters taboo words that, if entered by both other players, gives the third player points. The ESP Game and its variants bear the most similarity to consensus agreement games in that they look for the same label produced by multiple players in an interactive setting. The difference between these games and consensus agreement games is that the latter allows players to see and respond to the labels provided by the others in their clique. This allows players to guide each other to the correct answer in groups whose social dynamics avoid many of the frailties found in real-world decision-making groups [78, 125].

Our work is inspired by that of [57, 61], who explored the ability for a group of individuals to solve graph consensus problems through limited local interaction as a function of the topology of the network connecting them. Our approach is different because we require a non-collaborative agreement between disjoint subgraphs and design financial incentives which drive players toward the correct consensus rather than just any consensus.

Finally, it has been shown that crowdsourcing outcomes can be improved if worker compensation depends on matching answers with those of one or more

other workers. [54, 32] show improved outcomes if bonuses are given for a worker matching that of another single worker. [105] showed similar improvements when bonuses are based on a worker matching the majority vote of a set of other workers, whereas [58, 19, 102] provide reward schemes that match a worker's answers to more complex functions of the answers of other workers. Unlike this previous work, we use agreement to determine if all the workers (within multiple collaborative consensus decision-making cliques) have converged to the same answer. Nonetheless, we seek agreement and could benefit from the forms of agreement explored in previous works with the caveat that since our goal is to label difficult-to-label samples, lessons learned about agreement from experiments on entire datasets may not apply.

3.3 Method

Our goal is to reduce MTurk error rate by having multiple workers interactively find a consensus for low-agreement samples. In the manner of [57] we formulate a consensus graph problem of $2N$ nodes organized into two disjoint cliques of N. The graph is solved when all $2N$ nodes are assigned the same label. The graph problem is made tractable by showing both cliques the same image. We hypothesize that this is sufficient information for the $2N$ players to solve the graph (based on the success of the ESP Game).

We explicate this pattern and describe how to implement it in the subsections below.

3.3.1 Consensus Agreement Games

Consensus agreement games (CAG) is an instantiation of the pattern described above. Namely, we take each clique as an N-way collaborative labeling game where the potential labels are constrained to the labels from a previous majority-vote labeling process plus enough random labels to make a set of K labels. K should be small so that we make full use of the information we gathered in the previous labeling process yet large enough so that cliques have a low probability of agreeing if the players collaboratively guess randomly. During the game a player can see the selections of the other $N - 1$ players and can freely change their own selection. No other interaction is allowed between players.

A game ends after a fixed time limit whereupon if all players have selected the same label then the game has reached a *consensus*. Two games, operated independently, make one CAG in which we look for *agreement* in the consensus outcomes.

A pair of games has four possible outcomes:

- Consensus agreement: both games achieve a consensus and they agree.
- Consensus disagreement: both games achieve a consensus but they disagree.
- Solitary consensus: one game achieves a consensus.
- No consensus: both games fail to achieve a consensus.

A label which achieves consensus agreement is deemed to be confident enough to be taken as an annotation for the sample.

Player labeling strategies will be determined by the game payoffs, P_i. We define three outcomes for a game (which is paired with a second game):

- Consensus agreement (P_1): a consensus is reached and it matches the consensus reached by the second game.
- Clique consensus (P_2): all players select the same label but it is not a consensus agreement.
- Discord (P_3): not all players select the same label.

We want to choose payoffs which support our goals. First, the payoff for a clique consensus must be higher than the payoff for discord. This incentivises the players to adopt a labeling strategy which is different from independent voting. However, players may adopt simple strategies to get a clique consensus (e.g., always follow the first person that votes). Therefore, the payoff for a consensus agreement must be higher than the payoff for a clique consensus. This incentivises the players to vote for the truth since they have no other way to interact with the second group of players. Thus, the payoffs must satisfy $P_1 > P_2 > P_3$.

Games have a fixed duration but not all images need the same amount of time to label. We use a 120 second timer and averaged out the needed time by packing 8 images into a single game. Accordingly, we created a payment schedule in quanta of $1/8$ cents where $P_1 = \$0.02125$, $P_2 = \$0.00625$, and $P_3 = \$0.00125$ so that the maximum payout per game is \$0.17 and the minimum payout is \$0.01. These values were selected based on the results of preliminary experiments.

3.4 Worker Experience

In this section we describe one of our experimental games from the perspective of an MTurk worker.

1. The HIT reward is $0.01 but the title advertises "(with bonuses)". The HIT description informs the worker that they will work "with other people". We require that workers have a 95% approval rate and at least 100 approved HITs.

2. A worker previewing the HIT is told that they will "play a 120 second game with other people" and the earnings are described as "If you and your group play well, you are able to each make up to $0.17 for 120 seconds of your time. This works out to $5.10 per hour. The base rate for your time is $0.01 for up to four minutes of your time. If your group agrees on the same label, then you will receive a bonus of $0.005 (each game consists of 8 labels, so up to an additional $0.04 per game). If your group agrees on the correct label, then you receive an additional $0.015 per label (up to $0.12 per game)."

3. After accepting a HIT the worker is presented with instructions on how to play the game, definitions of the categories, and information on common mistakes. In particular, they are told "You are allowed and encouraged to change your vote as you change your opinion of what the material is.", "You will be able to see, in real time, the choice of the other players in their own rows." and "At the end of the game, you want your votes to all be the same, if a consensus is reached then you are given a bonus. Remember you get a bigger bonus if you all choose the same label and the label is correct."

4. When the worker presses a button to indicate they are ready to play then they

are placed on a game queue. The worker is told that they are waiting for people to join and that "If you wait for 3 minutes and your game doesn't start, then you will get money if you stay on the webpage and submit the HIT when you are instructed to submit it."

If the worker waits for 3 minutes then they are moved into the exit survey directly and will receive the HIT reward of \$0.01 (which is $8 \times P_3$) for their time.

5. During a game a worker is told "You can change your vote as many times as you want. Remember you get a bigger bonus if everyone picks the same label for each pair of images and the label is correct." Below this the worker is shown 8 pairs of images. Each pair is a crop of the sample to be labeled, a crop of the entire picture and buttons indicating the current vote of each player (Figure 3.1). Clicking on a crop shows a higher resolution version. They are also shown the current votes from each player for each sample.

Below the final pair the worker is told "As long as the game is running, the other players can change their votes. You may want to change your vote depending on what the others players do." At the bottom of the page the 8 pairs are repeated with much smaller images and the same vote buttons (the same as Figure 3.1 except the images are 85% smaller). This compact summary lets the worker view votes and vote without having to scroll the page excessively.

The time remaining (updated each second) is displayed at the top, middle and bottom of the page. When the game ends each worker is sent to an individual exit survey.

6. On the exit survey page the worker is told "You have earned at least \$0.XX and will receive more if your group agrees with the second group. You must

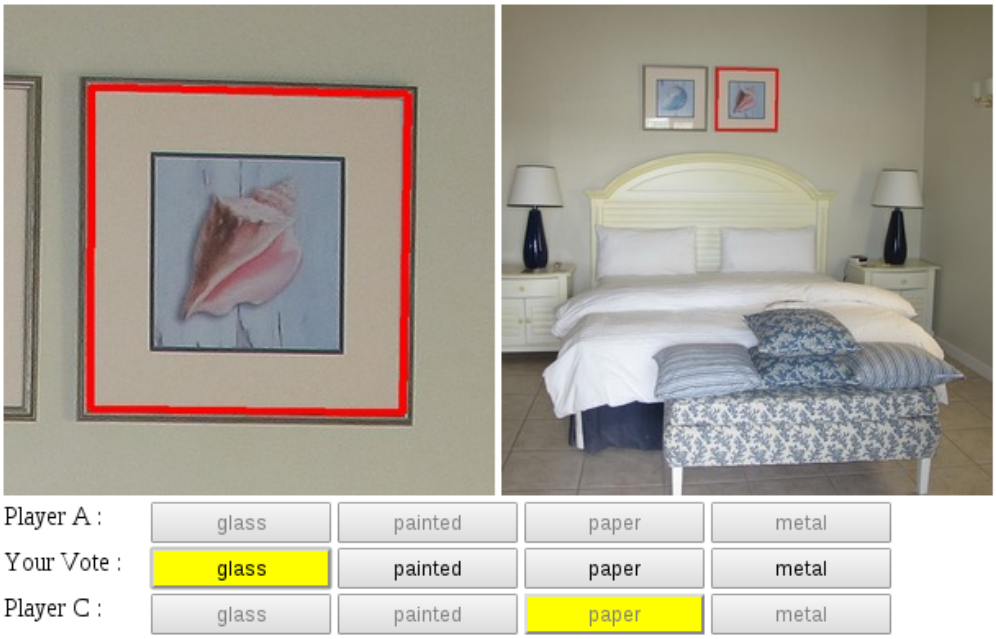


Figure 3.1: The collaborative labeling interface. **Top:** The material shape to be labeled is outlined in red. Clicking on either image will show a higher resolution image. The left image shows a crop of the shape, the right image shows where the shape appears in the photograph. **Bottom:** Buttons indicate the current selection of each player. Here, Player A has not yet made a selection, Player B (the current player) has selected glass and Player C has selected paper. Players may change their selection at any time, and the other players will see updates in real-time, but once a player has made a selection it is not possible for that player to return to the initial 'no-selection' state.

press 'Submit HIT with bonuses' to receive all the money you have earned (may take 48 hours)." where XX is 1 cent plus 1/2 cent per clique consensus. The worker is given the opportunity to provide optional feedback. We ask: "Do you have any feedback about the image labeling task?" and "Do you have any feedback about the webpage or game?"

3.5 Experimental Datasets

We want to evaluate CAG on natural images which are difficult to label yet not ambiguous. In this section we describe how we prepared two datasets which fit these criteria.

3.5.1 MINC

One dataset is the Materials in Context (MINC) Database, introduced in Chapter 2. We choose this dataset since it has 5 votes per sample and the data is hard to classify while requiring only an understanding of everyday materials. We define low-agreement samples as those which have exactly 3 out of 5 votes in agreement (in the future this definition could be expanded to include samples with even lower agreement among workers). This definition matches the experimental settings of MINC which defined high-agreement as four matching votes. See Figure 3.2 for examples.

We need ground truth to compute an error rate but ground truth is not available since the samples came from Internet photographs. Instead, we use experts to create a high-quality *expert truth* and rank different methods by comparing worker error rates against expert truth.

Three experts examined random low-agreement samples from the 10 largest categories of MINC and assigned annotations to samples which were unambiguous. In total, the experts annotated 456 samples with expert truth. The experts labeled as closely to truth as they could, and so were not limited to the 10 largest categories. Thus, we ended up with more than 10 categories.

Figure 3.2: Examples of low-agreement samples in MINC. **Top-Left:** This bowl received 3 votes for ceramic and 2 votes for glass. **Top-Right:** This pig received 3 votes for ceramic, 1 vote for plastic and 1 vote for foliage. **Bottom-Left:** This bottle received 3 votes for glass and 2 votes for plastic. **Bottom-Right:** This door received 3 votes for painted and 2 votes for wood.

3.5.2 Places

The MIT Places Database[1] [157] is a collection of images categorized by scene type. Since we do not know a priori which samples are difficult, we first selected 12 categories in 5 mutually confusing groups and collected 5 votes for 2400 images, 200 from each category. We then looked at the samples which received 3 of 5 agreement and assigned expert truth. Two of the mutually confusing groups had sufficient samples for experimentation. In order to prevent workers from being biased we randomly subsampled the largest categories so that no category was more than twice the smallest mutually confusing category. This rule only applied to the hotel room category, which was significantly over-represented

[1] http://places.csail.mit.edu/

since many bedrooms were actually hotel rooms.

We ended up with a low-agreement dataset of 22 *bedroom*, 32 *hotel room*, 16 *nursery* (the preceding constitute one mutually confusing group), 34 *coast*, 44 *ocean* and 28 *river* (the second mutually confusing group) images.

3.6 Experiments

In this section we evaluate performance. We first clarify the terminology of *labels* and *annotations*. A label is assigned to a sample by a worker. An annotation is assigned to a sample by a method. Not all samples receive an annotation. For example, if 5 workers label a sample with 5 unique labels then majority vote does not assign an annotation since no label achieved a majority.

Baseline method. We want to compare the MTurk error rate of CAG against a baseline. We selected majority vote of 7 unique workers (Vote7) as a baseline since majority vote is commonly used in practice and 7 voters prevents ties and is nearly the same number of people as in CAG when $N = 3$. We cannot use the original votes for the baseline since those workers chose from more than 4 possible choices (34 choices for MINC and 12 for Places). Instead, we collected new votes using the same protocol as CAG (4 choices, the original votes plus random labels). We set the per-label cost to \$0.004 and this decision was guided by the cost of MINC annotations (reported as \$0.0033 in Table 1 of [5]).

CAG settings. We took $K = 4$ so that the chance of random agreement is low. The clique size is a free parameter. We experimented with $N = 2$ and $N = 3$ since

the smallest possible clique will be the most cost effective and an odd-sized clique can prevent stalemates due to ties. We did not experiment with larger N since the cost per annotation would be too high. We report performance for both values of N on both datasets in Table 3.1 but for brevity we report results only for the best settings ($N = 2$ for MINC and $N = 3$ for Places) in the remainder of this section.

Comparison statistics. If one were labeling a dataset then two cliques would label each sample. However, this natural experiment would give us very little data for computing error rate. Instead we showed each sample to 3 cliques and formed all possible pairings. This gave us 3 times as much data for only 50% more experimental cost. We then used bootstrap sampling (1000 trials) to estimate performance statistics and standard errors (SE). Bootstrap sampling was not needed for the baseline method since Vote7 annotated a high-fraction of samples.

The MTurk error rate is (1 − Precision) as defined in Equation 3.1. For each method the true positives are annotations which match the expert truth and the false positives are those which do not.

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive + False Positive}} \tag{3.1}$$

We also report cost per annotation and *throughput* (the fraction of samples which receive an annotation). CAG throughput is reduced for three reasons: a player does not cast a vote (because they abandon the game or run out of time), players disagree or cliques disagree. We remind the reader that our goal is to augment an already large dataset with correctly annotated hard-to-label samples. Thus, CAG prioritizes accuracy over throughput which leads to higher

cost per annotation.

MTurk error rate. We find that CAG has a lower error rate (by at least 3 SE) than Vote7 on both datasets. See Table 3.1 for a summary. For MINC we had 456 low-agreement samples of which 427 received a Vote7 annotation for a throughput of 0.9364. With 344 correct annotations the MTurk error rate is 0.1944. CAG has an estimated MTurk error rate of 0.1186 with standard error 0.01257 and throughput of 0.4997 with SE 0.01335. For Places we had 176 low-agreement samples of which 170 received a Vote7 annotation for a throughput of 0.9659. With 130 correct annotations the MTurk error rate is 0.2353. CAG has an estimated MTurk error rate of 0.1490 with SE 0.02493 and throughput of 0.3842 with SE 0.02216.

Cost per annotation. The cost of a Vote7 annotation is determined by throughput and the reward of \$0.20 per HIT of 50 images which gives a cost of $7 \times \$0.004/0.9364 = \$0.0299$ per MINC annotation and \$0.0290 per Places annotation. For CAG the costs are variable. We pay workers for their time (\$0.01 per game), a clique for achieving a consensus (\$0.005 each) and a pair of cliques for achieving a consensus agreement (\$0.015 each). Our estimated cost per MINC annotation is \$0.104, SE \$0.00105. The breakdown is 9% for worker time, 33% for clique consensus bonuses, and 58% for consensus agreement bonuses. Our estimated cost per Places annotation is \$0.168, SE \$0.00392. The breakdown is 12% for worker time, 35% for clique consensus bonuses, and 54% for consensus agreement bonuses. Since the costs for each dataset are similar we report the combined cost in Table 3.1.

Acquiring more votes. We hypothesized that since we specifically selected samples which are known to be labeled with low confidence (and empirically found to have low accuracy) by an independent worker method then acquiring even more independent labels would not ultimately converge to the correct annotation. To test this hypothesis we used the baseline method to gather 21 votes per sample for the MINC dataset and the measured error rates at 7, 11 and 21 votes are 0.1944, 0.1900 and 0.1962, respectively. This confirms that for low-agreement samples additional independent votes does not converge to lower error rates.

High-agreement samples. We hypothesized that CAG would work best on low-agreement samples. For a comparison, we ran CAG and Vote7 on 352 MINC samples with 4 out of 5 agreement. The Vote7 MTurk error rate is 0.0698 with a throughput of 0.9773. The estimated CAG error rate is 0.05603 with SE 0.009223 and throughput of 0.5901 with SE 0.01496. The CAG mean is lower but within 1.5 SE so we find little difference between the two methods.

Consensus game. How important is agreement for CAG? We can conduct a single consensus game (CG) without looking for agreement between two cliques. This is attractive since the cost per annotation can be halved. The obvious downside is that there is no longer a financial incentive for the clique to label the image correctly.

For this method there are only two payoffs and they must satisfy $P_2 > P_3$. In our experiments we use P_2 = \$0.02125 and P_3 = \$0.00125 so that the pay per worker is the same as the consensus agreement games but the cost per annotation is approximately half.

Method	MINC	Places	Cost
Vote7	0.1944	0.2353	$0.0296
CAG N=2	**0.1186**	0.2407	$0.1043
CAG N=3	0.1410	**0.1490**	$0.1685

Table 3.1: MTurk error rate for low-agreement samples and cost per annotation. Consensus agreement games (CAG) reduces the error rate. The clique size (N) which gives the best performance depends on the dataset although $N = 3$ outperforms the baseline on both datasets.

We tested CG on MINC and found it much worse than Vote7. The MTurk error rate is 0.2730 and the throughput is 0.6667. Clearly workers found the HITs very lucrative since they enthusiastically snapped up CG HITs as soon as we posted them. We find that consensus does not perform well but it becomes a useful mechanism when paired with agreement.

Agreement incentives. How important is the clique consensus compared to the incentives for agreement? We conducted experiments on MINC which included the agreement incentives but excluded the collaborative clique game. This experiment's HITs are similar to a Vote7 HIT but the payment schedule is $0.01 for the worker's time and $0.02 per annotation (two workers agree). HITs contain 20 samples and have a time limit of 15 minutes. We found that the estimated MTurk error rate is 0.2054 with SE 0.01401 and throughput is 0.5904 with SE 0.01353 which makes this method on par with Vote7 (within 0.8 SE). Although peer incentives are effective, they become even more effective when combined with consensus games in our setting of difficult-to-label samples.

3.7 Analysis

We want to look more deeply into how the workers played the game so we instrumented some MINC CAG games on low-agreement samples and recorded all moves made by the players. We used $N = 3$ since larger cliques may have more interesting behaviors and we used MINC since it has more categories. There are two instrumented games per sample so the game pairs were used directly without the data augmentation described in the previous section.

Based on preliminary experiments we gave players 120 seconds to label 8 images. Yet, we found that the mean time of the last player activity was 77 seconds with deviation 20. This indicates players were under some time pressure and increasing the time allotted may reduce error further.

Players are allowed to change their labels so we looked for changes in the 2568 labelings. We found that in 186 cases the worker changed their label and in 162 of those cases the final label was different from the initial label.

How many different labels get considered as potential annotations for a sample? In Figure 3.3 we look at a histogram of the number of different labels a worker considers for a sample. In 2382 cases workers selected one label, a worker selected two different labels 178 times, three different labels 7 times and all four possible labels once. We see that most players do not change their label (i.e., their first label is final) and when they do change their label they very rarely pick a third or fourth choice. What about a clique? As we can see, a sample receives more different votes from a clique than it does from just one worker. This implies that a clique does increase the amount of knowledge which is applied to a sample. How about across both cliques? Although the cliques cannot

communicate directly we can still see that the label space is explored even more by 6 workers. These observations indicate the not all workers have the same knowledge. Each person is bringing their own opinions and sharing them with their clique.

Does this extra knowledge from fellow clique members help or hinder? We looked at how often an initially incorrect label was corrected and vice versa. We found that 100 initially incorrect labels were corrected and 44 initially correct labels were misled. This indicates that the cliques help guide members to the correct answer more often than they drive them away.

We know that players changed their labels, but how often do they do it? For example, do they change from A to B then back to A? If each change was decisive then there would have been 195 changes — the minimum number of changes needed (which was computed from the number of unique labels per sample made by players). We found that players changed their labels 230 times, therefore there were instances where players switched their labels more often than necessary. This could have been attempts to signal other players and/or it could be evidence of indecisiveness.

3.8 Observations About Workers

MTurk workers seemed to enjoy the game. We got lots of positive feedback: "Thanks, this was pretty fun and different way of doing it!", "It was my first time playing game like this. Much better than simple labeling", "Fun game! Well done HIT!", "That was fun!", "enjoyed a lot", and "very cute".

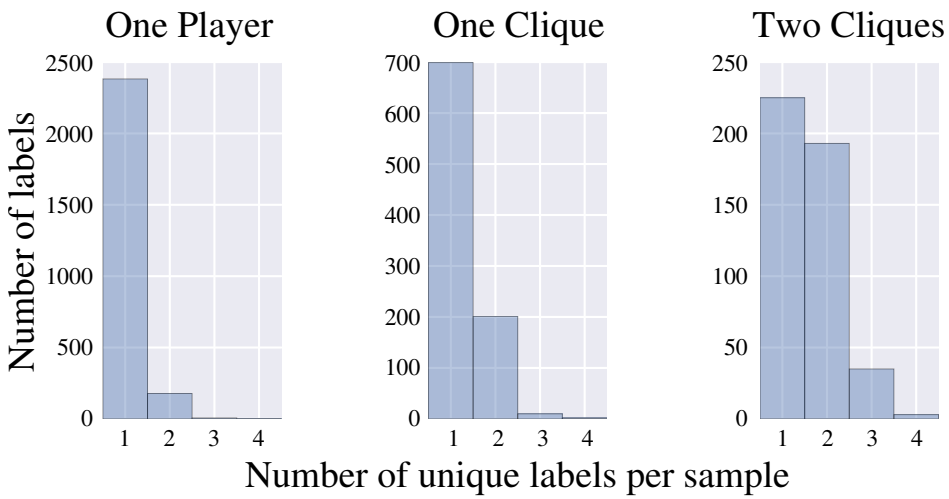


Figure 3.3: **Left:** The number of different choices made for a sample by a single player. **Middle:** The number of different choices made for a sample by a single clique. **Right:** The number of different choices made for a sample by both cliques. Single workers consider a small number of choices whereas the cliques increase the variety of choices considered for annotation.

Workers commented on the collaborative aspects of the game: "Wish we could post a message to other players in real time. I didn't change one of my labels because I thought I had a good reason for doing so.", "I feel like the second to last was hard, it was the one my group did not agree on. It was painted, yes, but painted wood.", "It's an interesting concept. I'm amazed my group mostly agreed.", "I'm not sure what to do if I think we labeled something wrong, but we marked it so we would all agree?", and "Would have been more interesting if you could play once alone, and then with the other and see if the players do any changes."

At least one worker had the disappointing experience of the other players abandoning the game: "I don't think it's fair that I only get a bonus if other people are paying attention. Neither of the other people in my group even did the task. It should be based on how many I got correct, not how many the other

people bothered to do.". One worker appears to have learned something about the other workers, namely that people often mislabel painted doors as wood: "make the directions about painted surfaces bold".

We had no problem getting workers to play our games. We could usually start one game per minute. It was observed that many workers would take another HIT and play subsequent games. This caused a phenomenon where small groups of workers may play several games together. In general, there was enough activity that workers played games rather than getting paid $0.01 for waiting 3 minutes.

One problem we encountered was that a small number of workers would take many HITs at once. Those workers learned not to enter the queue more than 2 or 3 times simultaneously because it is difficult to play multiple games at once. However, by holding the HITs they prevented other players from joining the queue so that games could not begin. We decided to allow multiple-queueing, but lowered the HIT duration so that a worker could only comfortably take 4 HITs at once before they had to begin returning HITs.

To better understand the workers the authors took the role of a player (in games not used for analysis or experimentation). We observed two interesting behaviors. In one case, the workers agreed that a sample was painted but we kept our vote as wallpaper (which was the correct answer). As we approached the end of the game one of the workers rapidly changed their vote between painted and wallpaper. They may have been trying to draw attention to the disagreement in the hopes of getting someone to change their vote. Or, they may have been trying to signal that they were indifferent to the two choices. The second interesting behavior was in a different game. In the beginning, the

third player did not cast any votes. This can happen if they are absent or have returned the HIT after joining the game queue. However, near the end of the game that third player suddenly cast a vote for every consensus that the two other players had established. We hypothesize that this player was following a strategy that maximized their payout without requiring their full attention — they simply let the other two players do all the work and copied their votes.

We anticipated that workers may try to collude to force a global consensus every time. This is hard since workers do not know when the second clique will run (or even if they are in the first or second clique) and our queueing system prevents any worker from the first clique entering the second clique. Nonetheless, it was suggested to the authors that workers could use a predetermined strategy of always picking the label which comes first lexicographically. We tried to oppose such strategies by including at least one random label in the set of candidates and shuffling the button order for each game.

In one case, we observed two workers that would very often play the game together. This could simply be because they enjoyed the game and were actively seeking our HITs. Another hypothesis is that this was the same person or two people in the same room taking control of the game by controlling two votes. We inspected the game records and found that these two players did not always agree and thus concluded that they were not colluding. However, we recommend that the queueing system prevent pairs of workers from entering the same games too often.

3.9 Discussion

This work has demonstrated that consensus agreement games can improve the annotations of images on which independent workers disagree. We have explored only two instances of this approach, and one can imagine varying design choices in this space to explore their impact on cost and effectiveness. For example, using cliques of size larger than three or varying the number of parallel cliques that must agree would impact the cost and accuracy of consensus agreement games. Indeed, while our results show that agreement is necessary, one could imagine doing so only at random on a subset of cases, to reduce cost while hopefully maintaining effectiveness. Our current approach only allows workers to change labels once the worker has provided a label, with no opportunity to "erase" the vote without picking an alternative. Allowing vote erasing, restricting the number of times an answer can be changed, or allowing workers to define arbitrary labels as in the ESP Game would impact cost and effectiveness in as yet unexplored ways.

There are also design choices relevant to the worker experience. Payment sizes, the number of images presented per game, amount of time given for each game, worker qualifications, throttling repeat players, and rules about simultaneous queueing and game playing could also all affect MTurk error rate and throughput. Results may be further improved by implementing cheating prevention mechanisms such as random clique assignment and blocking previously seen worker pairings. Worker attention may be improved by adding sentinel samples. Since workers seem to enjoy playing it may not be necessary to pay for the time workers wait on the game queue. Simply allowing the worker to return a HIT if they tire of waiting would simplify the game logic and

lower costs. Worker enthusiasm can also be taken as a sign that rewards can be reduced even further. It is not possible to pay fractional cents so there is a limit to how low payoffs can go, and while we combine multiple images into each game, there are attentional limits to increasing that number too high. One way to reduce rewards is to allow workers to chain multiple games and combine their rewards into a single payout.

Whereas [57, 61] explored how a group of individuals reach consensus as a function of the topology of the network connecting them, cliques are one case in the space of networks — consensus agreement games could be based on other network structures. We followed Judd et al's approach of minimal communication channel amonst workers, but given the value that social transparency [53] and larger-scale communication [88] can have on online group performance, the nature of the communication allowed between workers could be expanded, such as letting a player highlight portions of images or providing a chatbox (as suggested by one worker). Given our observation of free riding in some game instances [73], mechanisms for player self-policing [69, 49, 28], such as allowing players to identify and remove idle players, might have value. Also, workers could become more effective consensus members if they are instructed to be open to different opinions, view conflict as natural, and avoid early decision making [39].

On the other hand, the social psychology literature has found a wide range of ways in which teams and groups do not work as well as we might expect [65, 78, 125], finding for example that group behavior can suffer from group polarization, in-group/out-group biases, social loafing, and amplification of individual cognitive biases. CAG avoids many of these issues, creating small

group structures that violate the premises of much of such work. For example, social influence effects have limited relevance when your teammates are anonymous and only together for a short time with limited means of communications. There is minimal opportunity to have differentiated member roles, to see the effects of a team leader, to consider time-varying group composition, to have intragroup conflict, or to see impacts of suboptimal work flows when all workers are given identical tasks and incentives, remain anonymous to each other, and are teamed for timescales measured in minutes or seconds. Consensus games have found a way to take what had previously been perceived as problematic issues for small groups and has instead harnessed them as an asset. Social forces that encourage conformity are known to damage group performance, serving as a barrier to getting a diversity of knowledge and approaches. We instead present group tasks that actually seek conformity, both within each collaborative labeling clique and by using payments that target conformity to a second game's outcomes. Indeed, consensus agreement games may turn group polarization into an asset.

Consensus games, nonetheless are a form of small group activity, and need not be immune from the known inefficiencies of small groups. For example, our observations suggest the presence of social loafing. More generally, consensus games can be studied from the lens of social psychology, exploring such questions as what communication patterns (as constrained as they are) impact outcomes? How does what we know about the effects of time scarcity on small groups apply here? How do anchoring, priming, and other effects impact outcomes, for example in terms of what sequence of games (images, teammates) a player is given? What are the effects of sex, age, and personality differences on group performance? How can group outcomes be improved through individual

performance feedback? Do forms of organizational learning occur, and if so are they helpful? Are there ways in which mutual presence of team members (most typically visual) can positively impact group performance as it does in other small group settings? The answers to such questions are not just academic. The selection of who should be teamed and how their efforts should be structured can be informed by such knowledge, so that rather than assembling anonymous workers, we would assemble teams of the right people given the right tasks in the right way [39].

Whether one uses independent workers, consensus agreement games, or other crowdsourcing approaches for annotating data, they can all be loosely seen as sampling from a large population of individuals and eliciting information from them—either independently or collaboratively—so as to approximate what the majority vote of the entire population would be. Thus, for example, one might ask if an image of a cat is "cute", where this judgment should reflect what a typical person might answer, yet we are attempting to answer this without sampling the entire population. There are a variety of approaches that have been taken to perform tasks of this sort [11, 19, 25, 30, 42, 43, 76, 89, 91], and they might similarly suggest methods for using interacting workers to label data. Further, while we have proposed a hard-coded pragmatic approach of starting with votes among independent workers and then turning to consensus agreement games when there is insufficient support for a label, one could consider explicitly reasoning over workflows incorporating these and other consensus-seeking approaches [116, 152, 141].

3.10 Conclusion

We introduce consensus agreement games—a method for refining the majority vote labels on hard-to-label samples. We demonstrated this method on two real-world image datasets and showed that error rate was on average 37.8% lower than a majority vote baseline. The cost per annotation is high, but the method does not need to be run on the entire dataset. We suggest the following procedure. First, label all data with cost-efficient independent labeling tasks. Next, divide the dataset into subsets based on estimated difficulty. Next, take a small part of each subset, annotate them with expert truth then run CAG with $N = 2$ and $N = 3$. Select the best performing N and compare the difference of error rates for CAG annotations and independent worker annotations against the expert truth. Finally, use CAG to fully annotate those subsets for which the error rate (or difference of error rates) is larger than some threshold. The threshold is determined by either estimating the value of reducing error rate or setting a tolerance for maximum error rate.

In this chapter we grounded the method in two practical, unrelated image labeling tasks that demonstrate its success in the setting for which it was conceived. Yet, the pattern is general and may prove useful for other application domains. We believe there is value in the future to explore the merit of consensus agreement games on human computation tasks outside of image labeling.

3.11 Impact

Our work in this chapter was published at HCOMP 2016 [132]. This work anticipates an increasing need for high-quality annotated image datasets. Indeed, there has been a growing need for high-quality data for structured urban environments. Recent works are focused on high-quality segmentations [17, 93] (at a high cost of 1.5 hours / image) and highly-localized lane markings and drivable areas [150] (while introducing novel annotation tools to reduce cost). Whether or not collaborative games can reduce the cost of acquiring such high-quality datasets remains to be seen.

CHAPTER 4

DEEP FEATURE INTERPOLATION

4.1 Introduction

Generating believable changes in images is an active and challenging research area in computer vision and graphics. Until recently, algorithms were typically hand-designed for individual transformation tasks and exploited task-specific expert knowledge. Examples include transformations of human faces [127, 62], materials [7, 1], color [153], or seasons in outdoor images [71]. However, recent innovations in deep convolutional auto-encoders [106] have produced a succession of more versatile approaches. Instead of designing each algorithm for a specific task, a conditional (or adversarial) generator [67, 44] can be trained to produce a set of possible image transformations through supervised learning [148, 137, 158]. Although these approaches can perform a variety of seemingly impressive tasks, in this chapter we show that a surprisingly large set of them can be solved via linear interpolation in deep feature space and may not require specialized deep architectures.

How can linear interpolation be effective? In pixel space, natural images lie on an (approximate) non-linear manifold [139]. Non-linear manifolds are locally Euclidean, but globally curved and non-Euclidean. It is well known that in pixel space linear interpolation between images introduces ghosting artifacts, a sign of departure from the underlying manifold, and linear classifiers between image categories perform poorly.

On the other hand, deep convolutional neural networks (convnets) are

Figure 4.1: Aging a face with DFI.

known to excel at classification tasks such as visual object categorization [120, 47, 52]—while relying on a simple linear layer at the end of the network for classification. These linear classifiers perform well because networks map images into new representations in which image classes are *linearly* separable. In fact, previous work has shown that neural networks that are trained on sufficiently diverse object recognition classes, such as VGG [120] trained on ImageNet [70], learn surprisingly versatile feature spaces and can be used to train linear classifiers for additional image classes. Bengio et al. [8] hypothesize that convnets linearize the manifold of natural images into a (globally) Euclidean subspace of deep features.

Inspired by this hypothesis, we argue that, in such deep feature spaces, some image editing tasks may no longer be as challenging as previously believed. We propose a simple framework that leverages the notion that in the right feature space, image editing can be performed simply by linearly interpolating between images with a certain attribute and images without it. For instance, consider the task of adding facial hair to the image of a male face, given two sets of images: one set with facial hair, and one set without. If convnets can be trained

to distinguish between male faces with facial hair and those without, we know that these classes must be linearly separable. Therefore, motion along a single linear vector should suffice to move an image from deep features corresponding to “no facial hair” to those corresponding to “facial hair". Indeed, we will show that even a simple choice of this vector suffices: we average convolutional layer features of each set of images and take the difference.

We call this method Deep Feature Interpolation (DFI). Figure 4.1 shows an example of a facial transformation with DFI on a 390×504 image.

Of course, DFI has limitations: our method works best when all images are aligned, and thus is suited when there are feature points to line up (e.g. eyes and mouths in face images). It also requires that the sample images with and without the desired attribute are otherwise similar to the target image (e.g., if the input is a Caucasian male then the other images should be Caucasian males).

However, these assumptions on the data are comparable to what is typically used to train generative models, and in the presence of such data DFI works surprisingly well. We demonstrate its efficacy on several transformation tasks commonly used to evaluate generative approaches. Compared to prior work, it is much simpler, and often faster and more versatile: It does not require re-training a convnet, is not specialized on any particular task, and it is able to process much higher resolution images. Despite its simplicity we show that on many of these image editing tasks it outperforms state-of-the-art methods that are substantially more involved and specialized.

4.2 Related Work

Probably the generative methods most similar to ours are [72] and [104], which similarly generate data-driven attribute transformations using deep feature spaces. We use these methods as our primary points of comparison; however, they rely on specially trained generative auto-encoders and are fundamentally different from our approach to learning image transformations. Works by Reed et al. [106, 107] propose content change models for challenging tasks (identity and viewpoint changes) but do not demonstrate photo-realistic results. A contemporaneous work [9] edits image content by manipulating latent space variables. However, this approach is limited by the output resolution of the underlying generative model. An advantage of our approach is that it works with pre-trained networks and has the ability to run on much higher resolution images. In general, many other uses of generative networks are distinct from our problem setting [44, 22, 154, 111, 95, 24, 29], as they deal primarily with generating novel images rather than changing existing ones.

Gardner et al. [36] edits images by minimizing the witness function of the Maximum Mean Discrepancy statistic. The memory needed to calculate the transformed image's features by their method grows linearly whereas DFI removes this bottleneck.

Mahendran and Vedaldi [86] recovered visual imagery by inverting deep convolutional feature representations. Gatys et al. [38] demonstrated how to transfer the artistic style of famous artists to natural images by optimizing for feature targets during reconstruction. Rather than reconstructing imagery or transferring style, we edit the content of an existing image while seeking to

preserve photo-realism and all content unrelated to the editing operation.

Many works have used vector operations on a learned generative latent space to demonstrate transformative effects [26, 104, 40, 145]. In contrast, we suggest that vector operations on a discriminatively-trained feature space can achieve similar effects.

In concept, our work is similar to [127, 37, 129, 64, 62] that use video or photo collections to transfer the personality and character of one person's face to a different person (a form of puppetry [124, 140, 66]). This difficult problem requires a complex pipeline to achieve high quality results. For example, Suwajanakorn et al. [127] combine several vision methods: fiducial point detection [147], 3D face reconstruction [126] and optical flow [63]. Our method is less complicated and applicable to other domains (e.g., product images of shoes).

While we do not claim to cover all the cases of the techniques above, our approach is surprisingly powerful and effective. We believe investigating and further understanding the reasons for its effectiveness would be useful for better design of image editing with deep learning.

4.3 Deep Feature Interpolation

In our setting, we are provided with a test image $\mathbf{x}$ which we would like to change in a believable fashion with respect to a given attribute. For example, the image could be a man without a beard, and we would like to modify the image by adding facial hair while preserving the man's identity. We further assume access to a set of *target* images *with* the desired attribute $\mathcal{S}^t = \left\{\mathbf{x}_1^t, ..., \mathbf{x}_n^t\right\}$ (e.g., *men*

with facial hair) and a set of *source* images *without* the attribute $\mathcal{S}^s = \left\{\mathbf{x}_1^s, ..., \mathbf{x}_m^s\right\}$ (e.g., *men without facial hair*). Further, we are provided with a pre-trained convnet trained on a sufficiently rich object categorization task—for example, the openly available VGG network [120] trained on ImageNet [108]. We can use this convnet to obtain a new representation of an image, which we denote as $\mathbf{x} \rightarrow \phi(\mathbf{x})$. The vector $\phi(\mathbf{x})$ consists of concatenated activations of the convnet when applied to image $\mathbf{x}$. We refer to it as the *deep feature representation* of $\mathbf{x}$.

Deep Feature Interpolation can be summarized in four high-level steps (illustrated in Figure 4.2):

- We map the images in the target and source sets $\mathcal{S}^t$ and $\mathcal{S}^s$ into the deep feature representation through the pre-trained convnet ϕ (e.g., VGG-19 trained on ILSVRC2012).
- We compute the mean feature values for each set of images, $\bar{\phi}^t$ and $\bar{\phi}^s$, and define their difference as the *attribute vector*

$$\mathbf{w} = \bar{\phi}^t - \bar{\phi}^s. \tag{4.1}$$

- We map the test image $\mathbf{x}$ to a point $\phi(\mathbf{x})$ in deep feature space and move it along the attribute vector $\mathbf{w}$, resulting in $\phi(\mathbf{x}) + \alpha\mathbf{w}$.
- We can reconstruct the transformed output image $\mathbf{z}$ by solving the reverse mapping into pixel space w.r.t. $\mathbf{z}$

$$\phi(\mathbf{z}) = \phi(\mathbf{x}) + \alpha\mathbf{w}. \tag{4.2}$$

Although this procedure may appear deceptively simple, we show in Section 4.3 that it can be surprisingly effective. In the following we will describe some important details to make the procedure work in practice.

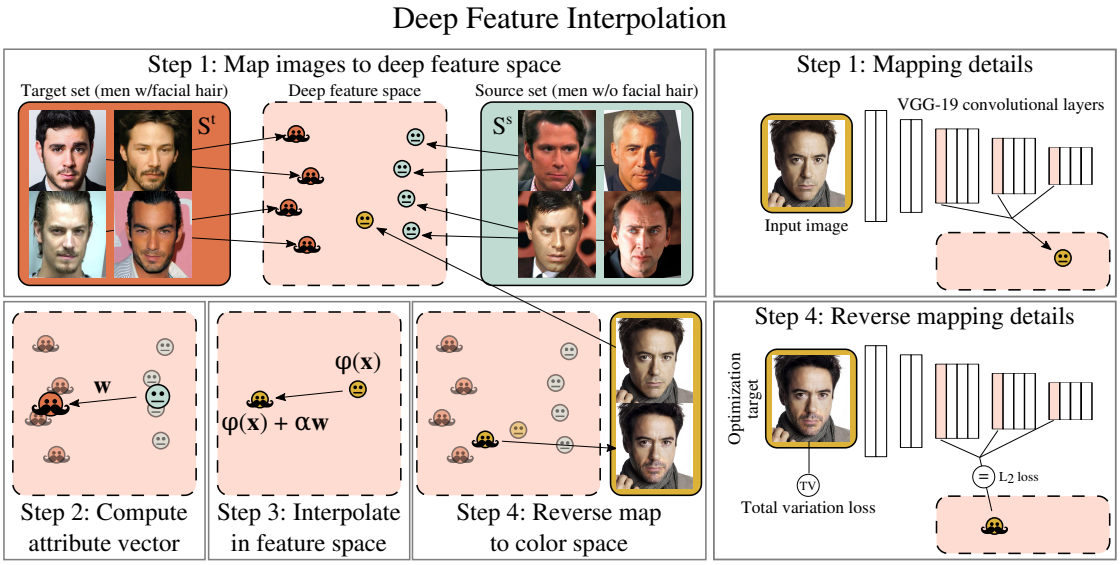


Figure 4.2: A schematic outline of the four high-level DFI steps.

Selecting $\mathcal{S}^t$ and $\mathcal{S}^s$. DFI assumes that the attribute vector $\mathbf{w}$ isolates the targeted transformation, i.e., it points towards the deep feature representation of image $\mathbf{x}$ with the desired attribute change. If such an image $\mathbf{z}$ was available (e.g., the same image of Mr. Robert Downey Jr. with beard), we could compute $\mathbf{w} = \phi(\mathbf{z}) - \phi(\mathbf{x})$ to isolate exactly the difference induced by the change in attribute. In the absence of the exact target image, we estimate $\mathbf{w}$ through the target and source sets. It is therefore important that both sets are as similar as possible to our test image $\mathbf{x}$ and there is no systematic attribute bias across the two data sets. If, for example, all target images in $\mathcal{S}^t$ were images of more senior people and source images in $\mathcal{S}^s$ of younger individuals, the vector $\mathbf{w}$ would unintentionally capture the change involved in aging. Also, if the two sets are too different from the test image (e.g., a different race) the transformation would not look believable. To ensure sufficient similarity we restrict $\mathcal{S}^t$ and $\mathcal{S}^s$ to the K nearest neighbors of $\mathbf{x}$. Let $\mathcal{N}_K^t$ denote the K nearest neighbors of $\mathcal{S}^t$ to $\phi(\mathbf{x})$;

we define

$$\bar{\phi}^t = \frac{1}{K} \sum_{\mathbf{x}^t \in \mathcal{N}_K^t} \phi(\mathbf{x}^t) \text{ and } \bar{\phi}^s = \frac{1}{K} \sum_{\mathbf{x}^s \in \mathcal{N}_K^s} \phi(\mathbf{x}^s)., \tag{4.3}$$

These neighbors can be selected in two ways, depending on the amount of information available. When attribute labels are available, we find the nearest images by counting the number of matching attributes (e.g., matching gender, race, age, hair color). When attribute labels are unavailable, or as a second selection criterion, we take the nearest neighbors by cosine distance in deep feature space.

Deep feature mapping. There are many choices for a mapping into deep feature space $\mathbf{x} \rightarrow \phi(\mathbf{x})$. We use the convolutional layers of the normalized VGG-19 network pre-trained on ILSVRC2012, which has proven to be effective at artistic style transfer [38]. The deep feature space must be suitable for two very different tasks: (1) linear interpolation and (2) reverse mapping back into pixel space. For the interpolation, it is advantageous to pick deep layers that are further along the linearization process of deep convnets [8]. In contrast, for the reverse mapping, earlier layers capture more details of the image [86]. The VGG network is divided into five pooling regions (with increasing depth). As an effective compromise we pick the first layers from the last three regions, `conv3_1`, `conv4_1` and `conv5_1` layers (after ReLU activations), flattened and concatenated. As the pooling layers of VGG reduce the dimensionality of the input image, we *increase* the image resolution of small images to be at least 200×200 before applying ϕ.

Image transformation. Due to the ReLU activations used in most convnets (including VGG), all dimensions in $\phi(\mathbf{x})$ are non-negative and the vector is sparse. As we average over K images (instead of a single image as in [8]), we expect

$\bar{\phi}^t, \bar{\phi}^s$ to have very small components in most features. As the two data sets $\mathcal{S}^t$ and $\mathcal{S}^s$ only differ in the target attribute, features corresponding to visual aspects unrelated to this attribute will be averaged to very small values and approximately subtracted away in the vector $\mathbf{w}$.

Reverse mapping. The final step of DFI is to reverse map the vector $\phi(\mathbf{x}) + \alpha\mathbf{w}$ back into pixel space to obtain an output image $\mathbf{z}$. Intuitively, $\mathbf{z}$ is an image that corresponds to $\phi(\mathbf{z}) \approx \phi(\mathbf{x}) + \alpha\mathbf{w}$ when mapped into deep feature space. Although no closed-form inverse function exists for the VGG mapping, we can obtain a color image by adopting the approach of [86] and find $\mathbf{z}$ with gradient descent:

$$\mathbf{z} = \arg\min_{\mathbf{z}} \frac{1}{2}\|(\phi(\mathbf{x}) + \alpha\mathbf{w}) - \phi(\mathbf{z})\|_2^2 + \lambda_{V^\beta} R_{V^\beta}(\mathbf{z}), \tag{4.4}$$

where R_{V^β} is the Total Variation regularizer [86] which encourages smooth transitions between neighboring pixels,

$$R_{V^\beta}(\mathbf{z}) = \sum_{i,j} \left((z_{i,j+1} - z_{i,j})^2 + (z_{i+1,j} - z_{i,j})^2 \right)^{\frac{\beta}{2}} \tag{4.5}$$

Here, $z_{i,j}$ denotes the pixel in location (i, j) in image $\mathbf{z}$. Throughout our experiments, we set $\lambda_{V^\beta} = 0.001$ and $\beta = 2$. We solve (4.4) with the standard hill-climbing algorithm L-BFGS [84].

4.4 Experimental Results

We evaluate DFI on a variety of tasks and data sets. For perfect reproducibility our code is available online[1].

[1] https://github.com/paulu/deepfeatinterp

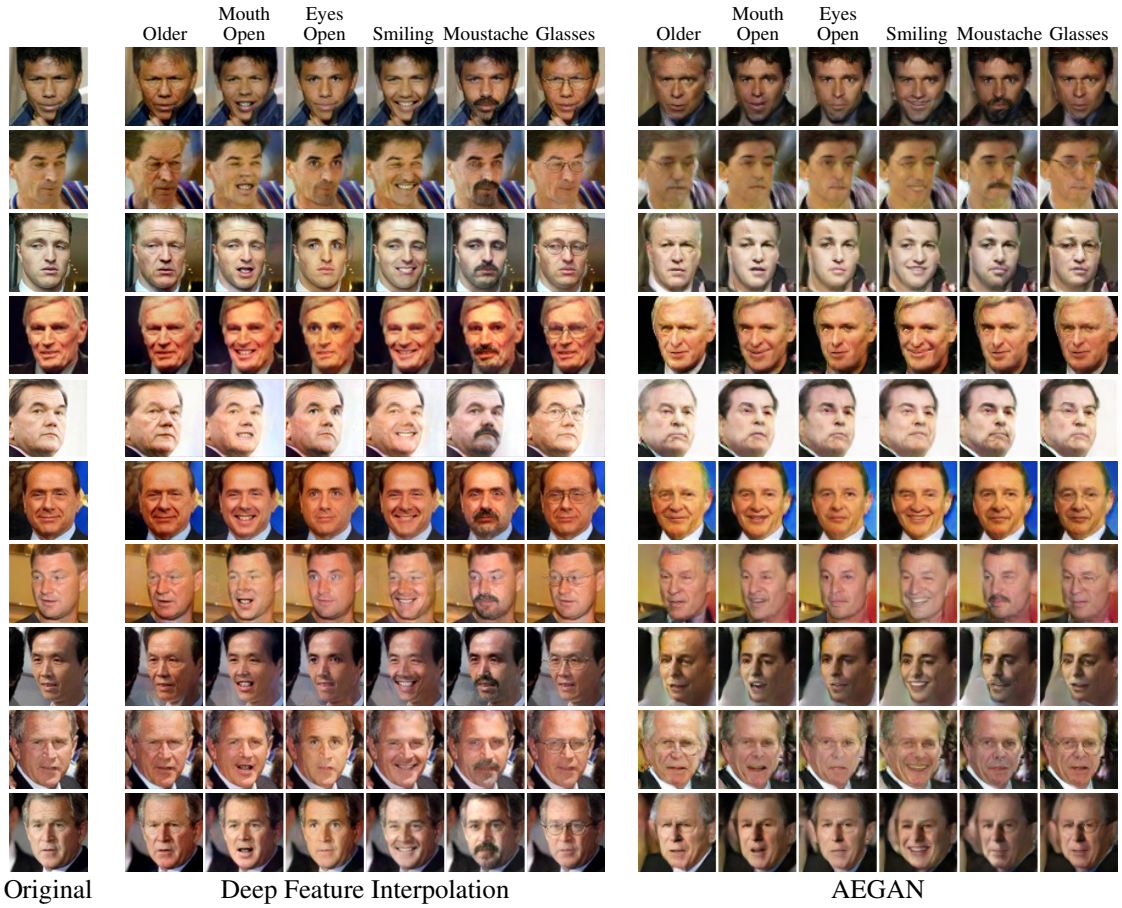


Figure 4.3: **(Zoom in for details.)** Adding different attributes to the same person (random test images). **Left.** Original image. **Middle.** DFI. **Right.** AEGAN. The goal is to add the specified attribute while preserving the identity of the original person. For example, when adding a moustache to Ralf Schumacher (3rd row) the hairstyle, forehead wrinkle, eyes looking to the right, collar and background are all preserved by DFI. No foreground mask or human annotation was used to produce these test results.

4.4.1 Changing Face Attributes

We compare DFI to AEGAN [72], a generative adversarial autoencoder, on several face attribute modification tasks. Similar to DFI, AEGAN also makes changes to faces by vector operations in a feature space. We use the Labeled Faces in the Wild (LFW) data set, which contains 13,143 images of faces with predicted annotations for 73 different attributes (e.g., SUNGLASSES, GENDER,

Figure 4.4: **(Zoom in for details.)** Filling missing regions. **Top.** LFW faces. **Bottom.** UT Zappos50k shoes. Inpainting is an interpolation from masked to unmasked images. Given any dataset we can create a source and target pair by simply masking out the missing region. DFI uses $K = 100$ such pairs derived from the nearest neighbors (excluding test images) in feature space. The face results match wrinkles, skin tone, gender and orientation (compare noses in 3rd and 4th images) but fail to fill in eyeglasses (3rd and 11th images). The shoe results match style and color but exhibit silhouette ghosting due to misalignment of shapes. Supervised attributes were not used to produce these results. For the curious, we include the source image but we note that the goal is to produce a plausible region filling—not to reproduce the source.

ROUND FACE, CURLY HAIR, MUSTACHE, etc.). We use these annotations as attributes for our experiments. We chose six attributes for testing: SENIOR, MOUTH SLIGHTLY OPEN, EYES OPEN, SMILING, MOUSTACHE and EYEGLASSES. (The negative attributes are YOUTH, MOUTH CLOSED, NARROW EYES, FROWNING, NO BEARD, NO EYEWEAR.) These attributes were chosen because it would be plausible for a single person to be changed into having each of those attributes. Our test set consists of 38 images that did not have any of the six target attributes, were not WEARING HAT, had MOUTH CLOSED, NO BEARD and NO EYEWEAR. As LFW is highly gender imbalanced, we only used images of the

more common gender, men, as target, source, and test images.

Matching the approach of [72], we align the face images and crop the outer pixels leaving a 100×100 face image, which we resize to 200×200. Target (source) collections are LFW images which have the positive (negative) attributes. From each collection we take the $K = 100$ nearest neighbors (by number of matching attributes) to form $\mathcal{S}^t$ and $\mathcal{S}^s$.

We empirically find that scaling $\mathbf{w}$ by its mean squared feature activation makes the free parameter somewhat more consistent across multiple attribute transformations. Let d be the dimensionality of $\phi(\mathbf{x})$ and define

$$\alpha = \frac{\beta}{\frac{1}{d}||\mathbf{w}||^2}. \tag{4.6}$$

We set $\beta = 0.4$.

Comparisons are shown in Figure 4.3. Looking down each column, we expect each image to express the target attribute. Looking across each row, we expect to see that the identity of the person is preserved. Although AEGAN often produces the right attributes, it does not preserve identity as well as the much simpler DFI.

Perceptual Study. Judgments of visual image changes are inherently subjective. To obtain an objective comparison between DFI and AEGAN we conducted a blind perceptual study with Amazon Mechanical Turk workers. We asked workers to pick the image which best expresses the target attribute while preserving the identity of the original face. This is a nuanced task so we required workers to complete a tutorial before participating in the study. The task was a forced choice between AEGAN and DFI (shown in random order) for six

older	mouth open	eyes open	smiling	moustache	glasses
4.57	7.09	17.6	20.6	24.5	38.3

Table 4.1: Perceptual study results. Each column shows the ratio at which workers preferred DFI to AEGAN on a specific attribute change (see Figure 4.3 for images).

attribute changes on 38 test images. We collected an average of 29.6 judgments per image from 136 unique workers and found that DFI was preferred to AEGAN by a ratio of 12:1. The least preferred transformation was Senior at 4.6:1 and the most preferred was Eyeglasses at 38:1 (see Table 4.1).

4.4.2 High Resolution Aging and Facial Hair

One of the major benefits of DFI over many generative models is the ability to run on high resolution images. However, there are several challenges in presenting results on high resolution faces.

First, we need a high-resolution dataset from which to select $\mathcal{S}^s$ and $\mathcal{S}^t$. We collect a database of 100,000 high resolution face images from existing computer vision datasets (CelebA, MegaFace, and Helen) and Google image search [85, 92, 74]. We augment existing datasets, selecting only clear, unobstructed, front-facing high-resolution faces. This is different from many existing datasets which may have noisy and low-resolution images.

Next, we need to learn the attributes of the images present in the face dataset to properly select source and target images. Because a majority of images we collect do not have labels, we use face attribute classifiers developed using labeled data from LFW and CelebA.

Finally, the alignment of dataset images to the input image needs to be as close as possible, as artifacts that result from poor alignment are more obvious at higher resolutions. Instead of aligning our dataset as a preprocessing step, we use an off-the-shelf face alignment tool in DLIB [60] to align images in $\mathcal{S}^s$ and $\mathcal{S}^t$ to the input image at test time.

We demonstrate results on editing megapixel faces for the tasks of aging and adding facial hair in Figures 4.5 and 4.6.

4.4.3 Inpainting Without Attributes

Inpainting fills missing regions of an image with plausible pixel values. There can be multiple correct answers. Inpainting is hard when the missing regions are large (see Figure 4.4 for our test masks). Since attributes cannot be predicted (e.g., eye color when both eyes are missing) we use distance in feature space to select the nearest neighbors.

Inpainting may seem like a very different task from changing face attributes, but it is actually a straightforward application of DFI. All we need are source and target pairs which differ only in the missing regions. Such pairs can be generated for any dataset by taking an image and masking out the same regions that are missing in the test image. The images with mask become the source set and those without the target set. We then find the $K = 100$ nearest neighbors in the masked dataset (excluding test images) by cosine distance in VGG-19 pool5 feature space. We experiment on two datasets: all of LFW (13,143 images, including male and female images) and the Shoes subset of UT Zappos50k (29,771 images) [149, 99]. For each dataset we find a single β that works well (1.6 for

Figure 4.5: **(Zoom in for details.)** Aging megapixel faces. **1st column.** Original image. **2nd and 3rd columns.** Two different aging intensities are shown ($\beta = \{0.15, 0.25\}$). Aging changes the skin—adding wrinkles and bags under the eyes.

Figure 4.6: **(Zoom in for details.)** Adding facial hair to megapixel faces. **1st column.** Original image. **2nd and 3rd columns.** Two different amounts of facial hair are shown ($\beta = \{0.15, 0.25\}$). Due to the step of our method which selects images with attributes similar to the input image, the young man (middle row) gains a dark-haired beard whereas the older man gains a salt-and-pepper beard. The result is both realistic and natural.

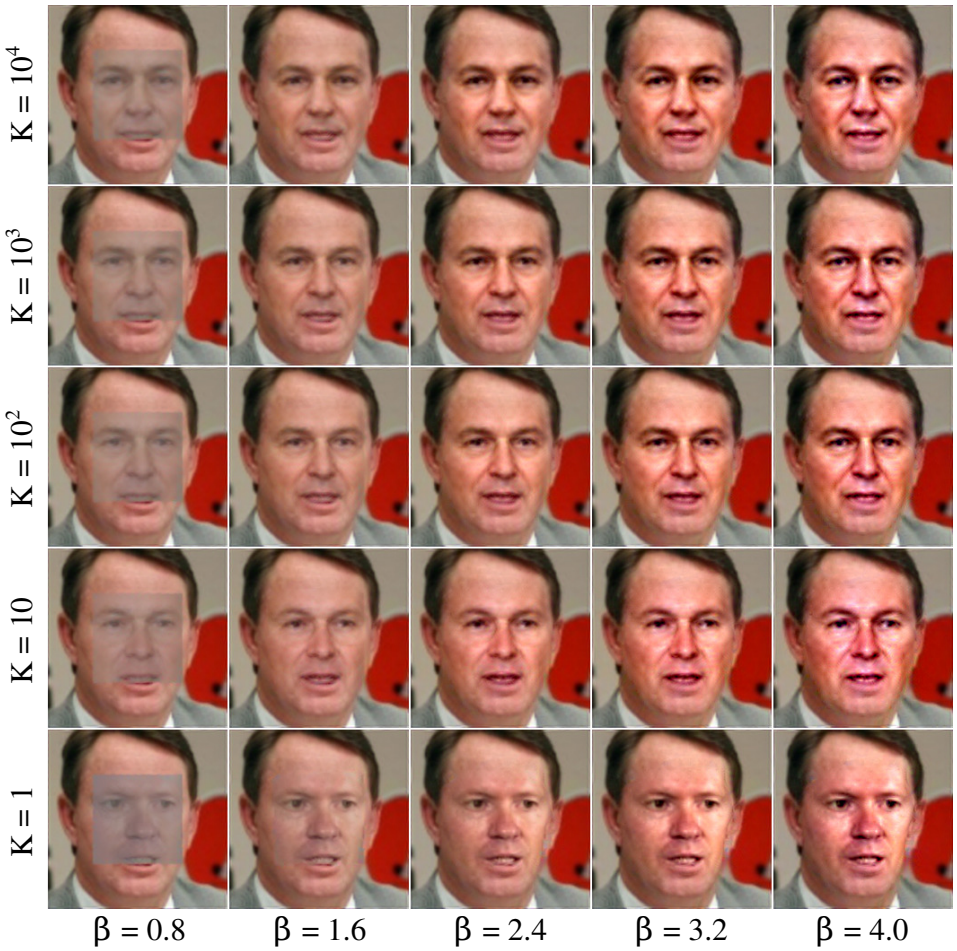


Figure 4.7: Inpainting and varying the free parameters. **Rows:** K, the number of nearest neighbors. **Columns:** β, higher values correspond to a larger perturbation in feature space. When K is too small the generated pixels do not fit the existing pixels as well (the nose, eyes and cheeks do not match the age and skin tone of the unmasked regions). When K is too large a difference of means fails to capture the discrepancy between the distributions (two noses are synthesized). When β is too small or too large the generated pixels look unnatural. We use $K = 100$ and $\beta = 1.6$.

LFW and 2.8 for UT Zappos50k).

We show our results in Figure 4.4 on 12 test images (more in supplemental) which match those used by disCVAE [148] (see Figure 6 of their paper). Qualitatively we observe that the DFI results are plausible. The filled face regions match skin tone, wrinkles, gender, and pose. The filled shoe regions match

Figure 4.8: Morphing a face to make it appear older. The transformation becomes more pronounced as the value of β increases.

color and shoe style. However, DFI failed to produce eyeglasses when stems are visible in the input and some shoes exhibit ghosting since the dataset is not perfectly aligned. DFI performs well when the face is missing (i.e., the central portion of each image) but we found it performs worse than disCVAE when half of the image is missing (Figure 4.11). Overall, DFI works surprisingly well on these inpainting tasks. The results are particularly impressive considering that, in contrast to disCVAE, it does not require attributes to describe the missing regions.

4.4.4 Varying the free parameters

Figure 4.7 illustrates the effect of changing β (strength of transformation) and K (size of source/target sets). As β increases, task-related visual elements change more noticeably (Figure 4.8). If β is low then ghosting can appear. If β is too large then the transformed image may jump to a point in feature space which leads to an unnatural reconstruction. K controls the variety of images in the source and target sets. A lack of variety can create artifacts where changed pixels do not match nearby unchanged pixels (e.g., see the lower lip, last row of Figure 4.7). However, too much variety can cause $\mathcal{S}^t$ and $\mathcal{S}^s$ to contain distinct subclasses and the set mean may describe something unnatural (e.g., in the first

row of Figure 4.7 the nose has two tips, reflecting the presence of left-facing and right-facing subclasses). In practice, we pick an β and K which work well for a variety of images and tasks rather than choosing per-case.

4.4.5 Making use of domain-specific knowledge

In this section we describe how we use preexisting knowledge about faces. We know that gender and expression can have a large impact on appearance. We also know that eyeglasses and hats are accessories that obscure parts of the face. These priors can be incorporated into DFI by modifying equation 4.3 so that N_K^t and N_K^s are restriced to images that do not have eyeglasses nor hats and match $\mathbf{x}$ in gender and $Smile$. These additional restrictions require a larger database with more variety in the specified attributes. Thus, we collected an additional 32k Internet images with relevant search terms: "man", "woman", "smile", "-smile". The effect of adding face-specific knowledge is an reduction of artifacts caused by averaging the various features of hats, eyeglasses, teeth, and hairstyle.

4.4.6 Domain transfer

We find that attribute vectors are surprisingly versatile. We can compute $\mathbf{w}$ on a database of human faces and transfer the visual effect of $\mathbf{w}$ to a different domain by replacing $\mathbf{x}$ with $\mathbf{x}'$ in Equation 4.4. In Figure 4.9 $\mathbf{x}'$ is a cat and the effect of adding facial hair, making the cat smile and adding glasses are plausible. It would be difficult to achieve these effects without domain transfer since images of smiling or bearded cats are not widely available.

Figure 4.9: **Domain transfer.** Attribute vectors are surprisingly versatile. We can compute an attribute vector using a database of *human* faces and apply it to a *cat* face. In this zero-shot out-of-domain setting, our method produces a plausible result.

4.4.7 Video

We can transform videos efficiently by using the preceeding frame as the initial solution to Equation 4.4. This approach reduces the number of optimization steps to 5 per video frame. The resulting video is smooth without temporal skips or texture motion (aka "swimming" or "crawling" textures).

4.4.8 Noise correction

The intention of $\mathcal{S}^s$ and $\mathcal{S}^t$ is that the difference of these sets succinctly captures the attribute change. In practice, the unpaired nature of photo collections means there will be noise in the attribute vector. The noise can be corrected by three steps: 1. establish explicit pairs across $\mathcal{S}^s$ and $\mathcal{S}^t$, 2. use image pairs to compute the importance of each element of $\mathbf{w}$, and 3. modify Equation 4.4 so that the more important elements of $\mathbf{w}$ receive more weight.

Step 1. The goal is to emulate an ideal paired dataset where the difference of an image pair includes only the attribute change. For each image in $\mathcal{S}^t$ we find the closest image in $\mathcal{S}^s$ in aligned, downsampled color space according to the distance metric

$$|| \downarrow (\mathbf{x}_i^s) - \downarrow (\mathbf{x}_j^t)||_2^2. \tag{4.7}$$

However, simply taking the nearest neighbor can concentrate source features because the same image could be selected multiple times. Instead, we greedily assign pairs so that no image is used more than once. Furthermore, we search over a larger set of $\mathcal{S}^s$ so that good pairs can be found.

Step 2. The most important elements of $\mathbf{w}$ are those which are consistent across all pairs computed in step 1. We will derive element-wise scalar importance weights from the variance. Let $\mathbf{w}_i$ be the difference of pair $\phi(\mathbf{x}_i^t)$ and $\phi(\mathbf{x}_i^s)$. First, we use the online Welford algorithm [142] to efficiently compute the element-wise variance

$$\mathbf{v}(j) = \frac{1}{n-1} \sum_{i=1}^{n} (\mathbf{w}_i(j) - \mathbf{w}(j))^2 \tag{4.8}$$

Next, we derive a *corrective weight vector* $\mathbf{c}$ parameterized by scalar s

$$\mathbf{c}(j) = \frac{n}{\sum_{i=1}^{n} \frac{1}{\mathbf{v}(i)+s}} \frac{1}{\mathbf{v}(j)+s} \tag{4.9}$$

The parameter s scales the effect of variance. DFI is equivalent to $s = \infty$. Empirically, we find that $s = 0.1$ works well for correcting noise.

Step 3. We account for feature importance by scaling the error terms of Equation 4.4 by $\mathbf{c}(i)$. The corrected reconstruction objective is

$$\mathbf{z} = \arg\min_{\mathbf{z}} \frac{1}{2} \sum_{i=1}^{n} ((\phi(\mathbf{x})(i) + \alpha \mathbf{w}(i) - \phi(\mathbf{z})(i))^2) + \lambda_{V^\beta} R_{V^\beta}(\mathbf{z}). \tag{4.10}$$

Figure 4.10: **Noise correction.** An input image (left) is made to grow a beard (top row) or look older (bottom row). The uncorrected noise of DFI leads to artifacts in the output image (middle). By introducing domain knowledge priors (Section 4.4.5) and noise correction (Section 4.4.8) the noise is greatly reduced.

Figure 4.10 shows the effect of noise correction. In the top row artifacts due to alignment are removed. In the bottom row the uncertainty over whether or not the hairline should recede is cleanly resolved. In addition, the area around the eyes and mouth is made clearer.

4.5 Discussion

In the previous section we have shown that Deep Feature Interpolation is surprisingly effective on several image transformation tasks. This is very promising and may have implications for future work in the area of automated image

transformations. However, DFI also has clear limitations and requirements on the data. We first clarify some of the aspects of DFI and then focus on some general observations.

Image alignment is a necessary requirement for DFI to work. We use the difference of means to cancel out the contributions of convolutional features that are unrelated to the attribute we wish to change, particularly when this attribute is centered in a specific location (adding a mustache, opening eyes, adding a smile, etc). For example, when adding a mustache, all target images contain a mustache and therefore the convolutional features with the mustache in their receptive field will not average out to zero. While max-pooling affords us some degree of translation invariance, this reasoning breaks down if mustaches appear in highly varied locations around the image, because no specific subset of convolutional features will then correspond to "mustache features". Image alignment is a limitation but not for faces, an important class of images. As shown in Section 4.4.2, existing face alignment tools are sufficient for DFI.

Time and space complexity. A significant strength of DFI is that it is very lean. The biggest resource footprint is GPU memory for the convolutional layers of VGG-19 (the large fully-connected layers are not needed). A 1280×960 image requires 4 GB and takes 5 minutes to reconstruct. A 200×200 image takes 20s to process. The time and space complexity are linear. In comparison, many generative models only demonstrate 64×64 images. Although DFI does not require the training of a specialized architecture, it is also fair to say that during test-time it is significantly slower than a trained model (which, typically, needs sub-seconds) As future work it may be possible to incorporate techniques from

real-time style-transfer [110] to speed-up DFI in practice.

DFI's simplicity. Although there exists work on high-resolution style transfer [38, 86, 110], to our knowledge, DFI is the first algorithm to enable automated high resolution content transformations. The simple mechanisms of DFI may inspire more sophisticated follow-up work on scaling up current generative architectures to higher resolutions, which may unlock a wide range of new applications and use cases.

Generative vs. Discriminative networks. To our knowledge, this work is the first cross-architectural comparison of an AE against a method that uses features from a discriminatively trained network. To our great surprise, it appears that a discriminative model has a latent space as good as an AE model at editing content. A possible explanation is that the AE architecture could organize a better latent space if it were trained on a more complex dataset. AE are typically trained on small datasets with very little variety compared to the size and richness of recognition datasets. The richness of ImageNet seems to be an important factor: in early experiments we found that the convolutional feature spaces of VGG-19 outperformed those of VGG-Face on face attribute change tasks.

Linear interpolation as a baseline. Linear interpolation in a pre-trained feature space can serve as a first test for determining if a task is interesting: problems that can easily be solved by DFI are unlikely to require the complex machinery of generative networks. Generative models can be much more powerful than linear interpolation, but the current problems (in particular, face attribute editing) which are used to showcase generative approaches are too simple. In-

Figure 4.11: Example of a hard task for DFI: inpainting an image with the right half missing.

deed, we do find many problems where generative models outperform DFI. In the case of inpainting we find DFI to be lacking when the masked region is half the image (Figure 4.11). DFI is also incapable of shape [159] or rotation [107] transformations since those tasks require aligned data. Finding more of these difficult tasks where generative models outshine DFI would help us better evaluate generative models. We propose DFI to be the linear interpolation baseline because it is very easy to compute, it will scale to future high-resolution models, it does not require supervised attributes, and it can be applied to nearly any aligned class-changing problem.

4.6 Conclusion

We have introduced DFI which interpolates in a pre-trained feature space to achieve a wide range of image transformations like aging, adding facial hair and inpainting. Overall, DFI performs surprisingly well given the method's simplicity. It is able to produce high quality images over a variety of tasks, in many cases of higher quality than existing state-of-the-art methods. This suggests that, given the ease with which DFI can be implemented, it should serve as a highly competitive baseline for certain types of image transformations on

aligned data. Given the performance of DFI, we hope that this spurs future research into image transformation methods that outperform this approach.

4.7 Impact

Our work in this chapter was published at CVPR 2017 [6]. This work has been used to improve recognition of occluded faces by generating same-identity faces given only the eyes [14], has led to a fast high-resolution portrait manipulation method [15] and has led to a real-time face editing app on a cellphone [94].

CHAPTER 5

CONCLUSION

Automatic scene understanding is a crucial component of building computer systems which can interact intelligently with the world. This thesis advances our knowledge of scene understanding in several directions.

Chapter 2 explores a new way to collect cost-efficient image annotations and a new method which combines a CNN and a dense CRF to infer pixel-level material categories given a single image. In Chapter 3 we develop a new collaborative labeling game which gathers high-accuracy image labels for scene understanding from crowdsource workers and we present new observations on crowdsource worker behaviour in anonymous small group activities. Chapter 4 presents new insights on the hidden organization of CNNs trained on a scene understanding task and a practical demonstration of how to apply those insights to believably edit the entangled visual structure of images.

Collectively these works increase our understanding of how to apply CNNs to recognize and edit a scene, how to gather data for training a scene understanding CNN and sheds light on why CNNs are good at scene understanding tasks.

5.1 Future Work

The deep learning approach to scene understanding has greatly improved performance across major scene understanding benchmarks and has unlocked previously intractable scene understanding applications. Yet, even as recogni-

tion rates approach human-level the task of semantic segmentation is far from solved. Scenes are composed of instances of *things* (e.g., people) and *stuff* (e.g., grass). Real-world scenes contain hundreds of instances so per-instance recognition rates must approach 99.99% accuracy in order to reduce scene-wide instance-level error below 1%. Even higher accuracies are needed to achieve less than 1% scene-wide error at the pixel level. Thus, characterizing and planning around the uncertainties of automatic predictions is a crucial part of deploying scene understanding methods in the real-world.

BIBLIOGRAPHY

[1] Miika Aittala, Timo Aila, and Jaakko Lehtinen. Reflectance modeling by neural texture synthesis. *ACM Transactions on Graphics (TOG)*, 35(4):65, 2016. 60

[2] Yoram Bachrach, Tom Minka, John Guiver, and Thore Graepel. How to grade a test without knowing the answers—a Bayesian graphical model for adaptive crowdsourcing and aptitude testing. In *International Conference on Machine Learning (ICML)*, 2012. 34

[3] Jeff Barr and Luis Felipe Cabrera. AI gets a brain. *ACM Queue*, 4(4):24–29, 2006. 32

[4] Amy Bearman, Olga Russakovsky, Vittorio Ferrari, and Li Fei-Fei. What's the point: Semantic segmentation with point supervision. In *European Conference on Computer Vision (ECCV)*, pages 549–565. Springer, 2016. 30

[5] Sean Bell, Paul Upchurch, Noah Snavely, and Kavita Bala. OpenSurfaces: A richly annotated catalog of surface appearance. *ACM Transactions on Graphics (TOG)*, 32(4), 2013. x, 8, 11, 13, 32, 44

[6] Sean Bell, Paul Upchurch, Noah Snavely, and Kavita Bala. Material recognition in the wild with the Materials in Context database. *Computer Vision and Pattern Recognition (CVPR)*, 2015. 30, 32, 85

[7] Rachele Bellini, Yanir Kleiman, and Daniel Cohen-Or. Time-varying weathering in texture space. *ACM Transactions on Graphics (TOG)*, 35(4):141, 2016. 60

[8] Yoshua Bengio, Grégoire Mesnil, Yann Dauphin, and Salah Rifai. Better mixing via deep representations. In *International Conference on Machine Learning (ICML)*, pages 552–560, 2013. 61, 67

[9] Andrew Brock, Theodore Lim, J. M. Ritchie, and Nick Weston. Neural photo editing with introspective adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2017. 63

[10] Barbara Caputo, Eric Hayman, and P. Mallikarjuna. Class-specific material categorisation. In *International Conference on Computer Vision (ICCV)*, pages 1597–1604, 2005. 10

[11] Ioannis Caragiannis, Ariel D. Procaccia, and Nisarg Shah. When do noisy votes reveal the truth? In *ACM Conference on Electronic Commerce*, pages 143–160, 2013. 57

[12] Kang Chen, Kun Xu, Yizhou Yu, Tian-Yi Wang, and Shi-Min Hu. Magic Decorator: automatic material suggestion for indoor digital scenes. *ACM Transactions on Graphics (TOG)*, 34(6):232, 2015. 31

[13] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, 2018. 31

[14] Xiang Chen, Linbo Qing, Xiaohai He, Jie Su, and Yonghong Peng. From eyes to face synthesis: a new approach for human-centered smart surveillance. *IEEE Access*, 6:14567–14575, 2018. 85

[15] Ying-Cong Chen, Huaijia Lin, Michelle Shu, Ruiyu Li, Xin Tao, Xiaoyong Shen, Yangang Ye, and Jiaya Jia. Facelet-bank for fast portrait manipulation. In *Computer Vision and Pattern Recognition (CVPR)*, pages 3541–3549, 2018. 85

[16] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Computer Vision and Pattern Recognition (CVPR)*, pages 3606–3613. IEEE, 2014. 11, 28, 29, 30

[17] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes dataset for semantic urban scene understanding. In *Computer Vision and Pattern Recognition (CVPR)*, pages 3213–3223, 2016. 59

[18] Kristin J Dana, Bram Van Ginneken, Shree K Nayar, and Jan J Koenderink. Reflectance and texture of real-world surfaces. *ACM Transactions on Graphics (TOG)*, 18(1):1–34, 1999. 10

[19] Anirban Dasgupta and Arpita Ghosh. Crowdsourced judgement elicitation with endogenous proficiency. In *International Conference on World Wide Web*, pages 319–330. ACM, 2013. 36, 57

[20] Ofer Dekel, Claudio Gentile, and Karthik Sridharan. Selective sampling

and active learning from single and multiple teachers. *The Journal of Machine Learning Research*, 13(1):2655–2697, 2012. 34

[21] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition (CVPR)*, pages 248–255. IEEE, 2009. 32, 34

[22] Emily L Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using a Laplacian pyramid of adversarial networks. In *Advances in Neural Information Processing Systems*, pages 1486–1494, 2015. 63

[23] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. DeCAF: A deep convolutional activation feature for generic visual recognition. In *International Conference on Machine Learning (ICML)*, pages 647–655, 2014. 28

[24] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. In *International Conference on Learning Representations (ICLR)*, 2017. 63

[25] Pinar Donmez, Jaime G. Carbonell, and Jeff Schneider. Efficiently learning the accuracy of labeling sources for selective sampling. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 259–268, 2009. 57

[26] Alexey Dosovitskiy, Jost Tobias Springenberg, and Thomas Brox. Learning to generate chairs with convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1538–1546, 2015. 64

[27] Fred Douglis. From the editor in chief: The search for Jim, and the search for altruism. *IEEE Internet Computing*, 11(3):4, 2007. 32

[28] Steven Dow, Anand Kulkarni, Scott Klemmer, and Björn Hartmann. Shepherding the crowd yields better work. In *Conference on Computer Supported Cooperative Work*, pages 1013–1022. ACM, 2012. 55

[29] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Alex Lamb, Martin Arjovsky, Olivier Mastropietro, and Aaron Courville. Adversarially learned inference. 2017. 63

[30] Seyda Ertekin, Haym Hirsh, and Cynthia Rudin. Learning to predict the wisdom of crowds. In *Conference on Collective Intelligence*, 2012. 57

[31] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision (IJCV)*, 88(2):303–338, June 2010. 14

[32] Boi Faltings, Radu Jurca, Pearl Pu, and Bao Duy Tran. Incentives to counter bias in human computation. In *AAAI Conference on Human Computation and Crowdsourcing (HCOMP)*, 2014. 36

[33] Clément Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1915–1929, 2013. 12

[34] Raghudeep Gadde, Varun Jampani, Martin Kiefel, Daniel Kappler, and Peter V. Gehler. Superpixel convolutional networks using bilateral inceptions. In *European Conference on Computer Vision (ECCV)*, pages 597–613. Springer, 2016. 31

[35] Yang Gao, Lisa Anne Hendricks, Katherine J Kuchenbecker, and Trevor Darrell. Deep learning for tactile understanding from visual and haptic data. In *International Conference on Robotics and Automation (ICRA)*, pages 536–543. IEEE, 2016. 31

[36] Jacob R Gardner, Paul Upchurch, Matt J Kusner, Yixuan Li, Kilian Q Weinberger, Kavita Bala, and John E Hopcroft. Deep Manifold Traversal: Changing labels with convolutional features. *arXiv preprint arXiv:1511.06421*, 2015. 63

[37] Pablo Garrido, Levi Valgaerts, Ole Rehmsen, Thorsten Thormahlen, Patrick Perez, and Christian Theobalt. Automatic face reenactment. In *Computer Vision and Pattern Recognition (CVPR)*, pages 4217–4224, 2014. 64

[38] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR)*, pages 2414–2423. IEEE, 2016. 63, 67, 83

[39] Martha E. Gentry. Consensus as a form of decision making. *Journal of Sociology & Social Welfare*, 9:233, 1982. 55, 57

[40] Rohit Girdhar, David F. Fouhey, Mikel Rodriguez, and Abhinav Gupta. Learning a predictable and generative vector representation for objects. In *European Conference on Computer Vision (ECCV)*, pages 484–499. Springer, 2016. 64

[41] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jagannath Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, pages 580–587. IEEE, 2014. 12

[42] Ashish Goel and David Lee. Triadic consensus. In *Internet and Network Economics*, pages 434–447. Springer, 2012. 57

[43] Ashish Goel and David T. Lee. Large-scale decision-making via small group interactions: the importance of triads. In *Workshop on Computational Social Choice (COMSOC)*, 2014. 57

[44] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014. 60, 63

[45] S. Gould, R. Fulton, and D. Koller. Decomposing a scene into geometric and semantically consistent regions. In *International Conference on Computer Vision (ICCV)*, 2009. 14

[46] Eric Hayman, Barbara Caputo, Mario Fritz, and Jan olof Eklundh. On the significance of real-world conditions for material classification. In *European Conference on Computer Vision (ECCV)*, 2004. 10

[47] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016. 61

[48] Chien-Ju Ho, Tao-Hsuan Chang, Jong-Chuan Lee, Jane Yung-jen Hsu, and Kuan-Ta Chen. KissKissBan: a competitive human computation game for image annotation. In *ACM SIGKDD Workshop on Human Computation*, pages 11–14, 2009. 35

[49] John J. Horton. Employer expectations, peer effects and productivity: Evidence from a series of field experiments. *Peer Effects and Productivity: Evidence from a Series of Field Experiments (August 3, 2010)*, 2010. 55

[50] Pei-Yun Hsueh, Prem Melville, and Vikas Sindhwani. Data quality from crowdsourcing: a study of annotation selection criteria. In *NAACL HLT Workshop on Active Learning for Natural Language Processing*, pages 27–35. Association for Computational Linguistics, 2009. 34

[51] Diane Hu, Liefeng Bo, and Xiaofeng Ren. Toward robust material recognition for everyday objects. In *British Machine Vision Conference (BMVC)*, pages 1–11. Citeseer, 2011. 7, 11, 12, 30

[52] Gao Huang, Zhuang Liu, Kilian Q. Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In *Computer Vision and Pattern Recognition (CVPR)*, 2017. 61

[53] Shih-Wen Huang and Wai-Tat Fu. Don't hide in the crowd!: increasing social transparency between peer workers improves crowdsourcing outcomes. In *SIGCHI Conference on Human Factors in Computing Systems*, pages 621–630. ACM, 2013. 55

[54] Shih-Wen Huang and Wai-Tat Fu. Enhancing reliability using peer consistency evaluation in human computation. In *Conference on Computer Supported Cooperative Work*, pages 639–648. ACM, 2013. 36

[55] Panagiotis G. Ipeirotis, Foster Provost, Victor S. Sheng, and Jing Wang. Repeated labeling using multiple noisy labelers. *Data Mining and Knowledge Discovery*, 28(2):402–441, 2014. 34

[56] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM International Conference on Multimedia*, pages 675–678, 2014. 22

[57] Stephen Judd, Michael Kearns, and Yevgeniy Vorobeychik. Behavioral dynamics and influence in networked coloring and consensus. *Proceedings of the National Academy of Sciences*, 107(34):14978–14982, 2010. 33, 35, 36, 55

[58] Ece Kamar and Eric Horvitz. Incentives and truthful reporting in consensus-centric crowdsourcing. Technical report, Technical report, MSR-TR-2012-16, Microsoft Research, 2012. 36

[59] Bob Kanefsky, Nadine G. Barlow, and Virginia C. Gulick. Can distributed volunteers accomplish massive data analysis tasks. *Lunar and Planetary Science*, 1, 2001. 32

[60] Vahid Kazemi and Josephine Sullivan. One millisecond face alignment with an ensemble of regression trees. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1867–1874, 2014. 73

[61] Michael Kearns. Experiments in social computation. *Communications of the ACM*, 55(10):56–67, 2012. 33, 35, 55

[62] Ira Kemelmacher-Shlizerman. Transfiguring portraits. *ACM Transactions on Graphics (TOG)*, 35(4):94, 2016. 60, 64

[63] Ira Kemelmacher-Shlizerman and Steven M. Seitz. Collection flow. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1792–1799. IEEE, 2012. 64

[64] Ira Kemelmacher-Shlizerman, Eli Shechtman, Rahul Garg, and Steven M. Seitz. Exploring photobios. In *ACM Transactions on Graphics (TOG)*, volume 30, page 61, 2011. 64

[65] Norbert L. Kerr and R. Scott Tindale. Group performance and decision making. *Annual Review of Psychology*, 55:623–655, 2004. 55

[66] Natasha Kholgade, Iain Matthews, and Yaser Sheikh. Content retargeting using parameter-parallel facial layers. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 195–204, 2011. 64

[67] Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pages 3581–3589, 2014. 60

[68] Philipp Krähenbühl and Vladlen Koltun. Parameter learning and convergent inference for dense random fields. In *International Conference on Machine Learning (ICML)*, pages 513–521, 2013. 9, 18, 20

[69] Robert E. Kraut, Paul Resnick, Sara Kiesler, Moira Burke, Yan Chen, Niki Kittur, Joseph Konstan, Yuqing Ren, and John Riedl. *Building successful online communities: Evidence-based social design*. MIT Press, 2012. 55

[70] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012. 12, 19, 21, 22, 28, 32, 61

[71] Pierre-Yves Laffont, Zhile Ren, Xiaofeng Tao, Chao Qian, and James Hays. Transient attributes for high-level understanding and editing of outdoor scenes. *ACM Transactions on Graphics (TOG)*, 33(4):149, 2014. 60

[72] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *International Conference on Machine Learning (ICML)*, 2016. 63, 69, 71

[73] Bibb Latane, Kipling Williams, and Stephen Harkins. Many hands make light the work: The causes and consequences of social loafing. *Journal of Personality and Social Psychology*, 37(6):822, 1979. 55

[74] Vuong Le, Jonathan Brandt, Zhe Lin, Lubomir Bourdev, and Thomas Huang. Interactive facial feature localization. *European Conference on Computer Vision (ECCV)*, pages 679–692, 2012. 72

[75] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989. 12

[76] David T. Lee, Ashish Goel, Tanja Aitamurto, and Helene Landemore. Crowdsourcing for participatory democracies: Efficient elicitation of social choice functions. In *AAAI Conference on Human Computation and Crowdsourcing (HCOMP)*, 2014. 57

[77] Thomas Leung and Jitendra Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *International Journal of Computer Vision (IJCV)*, 43(1):29–44, June 2001. 10

[78] John M. Levine and Richard L. Moreland. *Small groups: key readings*. Psychology Press, 2008. 35, 55

[79] David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In *ACM SIGIR International Conference on Research and Development in Information Retrieval*, pages 3–12. Springer-Verlag, 1994. 33

[80] Wenbin Li and Mario Fritz. Recognizing materials from virtual examples. In *European Conference on Computer Vision (ECCV)*, pages 345–358. Springer, 2012. 11

[81] Chien-Wei Lin, Kuan-Ta Chen, Ling-Jyh Chen, Irwin King, and Jimmy H. M. Lee. An analytical approach to optimizing the utility of ESP games. In *International Conference on Web Intelligence and Intelligent Agent Technology*, volume 1, pages 184–187. IEEE, 2008. 35

[82] Chris J. Lintott, Kevin Schawinski, Anže Slosar, Kate Land, Steven Bamford, Daniel Thomas, Jordan Raddick, Robert C. Nichol, Alex Szalay, Dan Andreescu, et al. Galaxy Zoo: morphologies derived from visual inspection of galaxies from the sloan digital sky survey. *Monthly Notices of the Royal Astronomical Society*, 389(3):1179–1189, 2008. 34

[83] Ce Liu, Lavanya Sharan, Edward H Adelson, and Ruth Rosenholtz. Exploring features in a Bayesian framework for material recognition. In *Computer Vision and Pattern Recognition (CVPR)*, pages 239–246. IEEE, 2010. 7, 11

[84] Dong C Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1-3):503–528, 1989. 68

[85] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *International Conference on Computer Vision (ICCV)*, 2015. 72

[86] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Computer Vision and Pattern Recognition (CVPR)*, 2015. 63, 67, 68, 83

[87] Andrew Mao, Ece Kamar, and Eric Horvitz. Why stop now? Predicting worker engagement in online crowdsourcing. In *AAAI Conference on Human Computation and Crowdsourcing (HCOMP)*, 2013. 34

[88] Andrew Mao, Winter Mason, Siddharth Suri, and Duncan J. Watts. An experimental study of team size and performance on a complex task. *PloS One*, 11(4):e0153048, 2016. 55

[89] Andrew Mao, Ariel D Procaccia, and Yiling Chen. Social choice for human computation. In *AAAI Conference on Human Computation and Crowdsourcing (HCOMP)*, 2012. 57

[90] Lucas Matney. Google has 2 billion users on Android, 500M on Google Photos. https://techcrunch.com/2017/05/17/google-has-2-billion-users-on-android-500m-on-google-photos/, 2017. [Online; accessed 15-June-2018]. 1

[91] Andrea Montanari and Amin Saberi. Convergence to equilibrium in local interaction games. In *IEEE Symposium on Foundations of Computer Science*, pages 303–312, 2009. 57

[92] Aaron Nech and Ira Kemelmacher-Shlizerman. Level playing field for million scale face recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2017. 72

[93] Gerhard Neuhold, Tobias Ollmann, S Rota Bulo, and Peter Kontschieder. The Mapillary Vistas dataset for semantic understanding of street scenes. In *International Conference on Computer Vision (ICCV)*, pages 22–29, 2017. 59

[94] Ransen Niu. Real-time image editing and iOS application with convolutional networks. Master's thesis, Cornell, Ithaca, New York, 2018. 85

[95] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier GANs. In *International Conference on Machine Learning (ICML)*, 2016. 63

[96] Union of Concerned Scientists contributors. UCS satellite database. https://www.ucsusa.org/nuclear-weapons/space-weapons/satellite-database, 2017. [Online; accessed 15-June-2018]. 1

[97] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1717–1724. IEEE, 2014. 12

[98] Aditya Parameswaran, Stephen Boyd, Hector Garcia-Molina, Ashish Gupta, Neoklis Polyzotis, and Jennifer Widom. Optimal crowd-powered rating and filtering algorithms. *Proceedings of the VLDB Endowment*, 7(9):685–696, 2014. 34

[99] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: Feature learning by inpainting. In *Computer Vision and Pattern Recognition (CVPR)*, pages 2536–2544, 2016. 73

[100] Genevieve Patterson, Chen Xu, Hang Su, and James Hays. The SUN attribute database: Beyond categories for deeper scene understanding. *International Journal of Computer Vision (IJCV)*, 108(1-2):59–81, 2014. 7

[101] Xianbiao Qi, Rong Xiao, Jun Guo, and Lei Zhang. Pairwise rotation invariant co-occurrence local binary pattern. In *European Conference on Computer Vision (ECCV)*, pages 158–171. Springer, 2012. 7, 11

[102] Goran Radanovic, Boi Faltings, and Radu Jurca. Incentives for effort in crowdsourcing using the peer truth serum. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 7(4):48, 2016. 36

[103] Jordan Raddick, C. J. Lintott, K Schawinski, Daniel Thomas, R. C. Nichol, D Andreescu, Steven Bamford, Kate R. Land, P Murray, A Slosar, Alexander S. Szalay, J Vandenberg, et al. Galaxy Zoo: an experiment in public science participation. In *Bulletin of the American Astronomical Society*, volume 39, page 892, 2007. 32

[104] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2016. 63, 64

[105] Huaming Rao, Shih-Wen Huang, and Wai-Tat Fu. What will others choose? How a majority vote reward scheme can improve human computation in a spatial location identification task. In *AAAI Conference on Human Computation and Crowdsourcing (HCOMP)*, 2013. 36

[106] Scott Reed, Kihyuk Sohn, Yuting Zhang, and Honglak Lee. Learning to disentangle factors of variation with manifold interaction. In *International Conference on Machine Learning (ICML)*, pages 1431–1439, 2014. 60, 63

[107] Scott E Reed, Yi Zhang, Yuting Zhang, and Honglak Lee. Deep visual analogy-making. In *Advances in Neural Information Processing Systems*, pages 1252–1260, 2015. 63, 84

[108] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 7, 12, 65

[109] Bryan C. Russell, Antonio Torralba, Kevin P. Murphy, and William T. Freeman. LabelMe: A database and web-based tool for image annotation. *International Journal of Computer Vision (IJCV)*, 77(1-3):157–173, May 2008. 14, 32

[110] Fereshteh Sadeghi, C Lawrence Zitnick, and Ali Farhadi. Visalogy: Answering visual analogy questions. In *Advances in Neural Information Processing Systems*, pages 1873–1881, 2015. 83

[111] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016. 63

[112] Carl Schissler, Christian Loftin, and Dinesh Manocha. Acoustic classification and optimization for multi-modal rendering of real-world scenes. *IEEE Transactions on Visualization and Computer Graphics*, 24(3):1246–1259, 2018. 31

[113] Gabriel Schwartz and Ko Nishino. Visual material traits: Recognizing per-pixel material context. In *International Conference on Computer Vision Workshops (ICCVW)*, pages 883–890. IEEE, 2013. 11

[114] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann LeCun. OverFeat: Integrated recognition, localization and detection using convolutional networks. In *International Conference on Learning Representations (ICLR)*. CBLS, April 2014. 9, 12, 20

[115] Nihar Bhadresh Shah and Denny Zhou. Double or nothing: Multiplicative incentive mechanisms for crowdsourcing. In *Advances in Neural Information Processing Systems*, pages 1–9, 2015. 34

[116] Dafna Shahaf and Eric Horvitz. Generalized task markets for human and machine computation. In *AAAI Conference on Artificial Intelligence*, 2010. 57

[117] Lavanya Sharan, Ce Liu, Ruth Rosenholtz, and Edward Adelson. Recognizing materials using perceptually inspired features. *International Journal of Computer Vision (IJCV)*, 2013. 7, 29

[118] Lavanya Sharan, Ruth Rosenholtz, and Edward Adelson. Material perception: What can you see in a brief glance? *Journal of Vision*, 9(8):784–784, 2009. 7, 11

[119] Victor S. Sheng, Foster Provost, and Panagiotis G. Ipeirotis. Get another label? Improving data quality and data mining using multiple, noisy labelers. In *SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 614–622. ACM, 2008. 32

[120] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015. 12, 21, 22, 61, 65

[121] Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Conference on Empirical Methods in Natural Language Processing*, pages 254–263. Association for Computational Linguistics, 2008. 32

[122] Alexander Sorokin and David Forsyth. Utility data annotation with Amazon Mechanical Turk. In *Computer Vision and Pattern Recognition Workshops*, pages 1–8. IEEE, 2008. 32

[123] Merrielle Spain and Pietro Perona. Some objects are more equal than others: Measuring and predicting importance. In *European Conference on Computer Vision (ECCV)*, pages 523–536. Springer, 2008. 32

[124] Robert W Sumner and Jovan Popović. Deformation transfer for triangle meshes. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 399–405, 2004. 64

[125] Cass R. Sunstein. *Infotopia: How many minds produce knowledge*. Oxford University Press, 2006. 35, 55

[126] Supasorn Suwajanakorn, Ira Kemelmacher-Shlizerman, and Steven M. Seitz. Total moving face reconstruction. In *European Conference on Computer Vision (ECCV)*, pages 796–812. Springer, 2014. 64

[127] Supasorn Suwajanakorn, Steven M. Seitz, and Ira Kemelmacher-Shlizerman. What makes Tom Hanks look like Tom Hanks. In *International Conference on Computer Vision (ICCV)*, pages 3952–3960, 2015. 60, 64

[128] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015. 12, 21, 22

[129] Justus Thies, Michael Zollhöfer, Matthias Nießner, Levi Valgaerts, Marc Stamminger, and Christian Theobalt. Real-time expression transfer for facial reenactment. *ACM Transactions on Graphics (TOG)*, 34(6):183, 2015. 64

[130] Rasmus Thogersen. Data quality in an output-agreement game: A comparison between game-generated tags and professional descriptors. In *Collaboration and Technology*, pages 126–142. Springer, 2013. 35

[131] Radu Timofte and Luc J Van Gool. A training-free classification framework for textures, writers, and materials. In *British Machine Vision Conference (BMVC)*, pages 1–12, 2012. 11

[132] Paul Upchurch, Daniel Sedra, Andrew Mullen, Haym Hirsh, and Kavita Bala. Interactive consensus agreement games for labeling images. In *AAAI Conference on Human Computation and Crowdsourcing (HCOMP)*, October 2016. 59

[133] Manik Varma and Andrew Zisserman. A statistical approach to texture classification from single images. *International Journal of Computer Vision (IJCV)*, 62(1-2):61–81, April 2005. 10

[134] Luis Von Ahn and Laura Dabbish. Labeling images with a computer game. In *SIGCHI Conference on Human Factors in Computing Systems*, pages 319–326. ACM, 2004. 32, 33, 35

[135] Byron C. Wallace, Kevin Small, Carla E. Brodley, and Thomas A. Trikalinos. Who should label what? Instance allocation in multiple expert active learning. In *SDM*, pages 176–187. SIAM, 2011. 34

[136] Ting-Chun Wang, Jun-Yan Zhu, Ebi Hiroaki, Manmohan Chandraker, Alexei A. Efros, and Ravi Ramamoorthi. A 4D light-field dataset and CNN architectures for material recognition. In *European Conference on Computer Vision (ECCV)*, pages 121–138. Springer, 2016. 31

[137] Xiaolong Wang and Abhinav Gupta. Generative image modeling using style and structure adversarial networks. In *European Conference on Computer Vision (ECCV)*, pages 318–335. Springer, 2016. 60

[138] Fabian L. Wauthier and Michael I. Jordan. Bayesian bias mitigation for crowdsourcing. In *Advances in Neural Information Processing Systems*, pages 1800–1808, 2011. 34

[139] Kilian Q Weinberger and Lawrence K Saul. Unsupervised learning of image manifolds by semidefinite programming. *International Journal of Computer Vision (IJCV)*, 70(1):77–90, 2006. 60

[140] Thibaut Weise, Hao Li, Luc Van Gool, and Mark Pauly. Face/off: Live facial puppetry. In *ACM SIGGRAPH/Eurographics Symposium on Computer animation*, pages 7–16, 2009. 64

[141] Daniel S. Weld. Intelligent control of crowdsourcing. In *International Conference on Intelligent User Interfaces*. ACM, 2015. 57

[142] BP Welford. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420, 1962. 80

[143] Andrew J. Westphal, Anna L. Butterworth, Christopher J. Snead, Nahide Craig, David Anderson, Steven M. Jones, Donald E. Brownlee, Richard Farnsworth, and Michael E. Zolensky. Stardust@Home: a massively distributed public search for interstellar dust in the stardust interstellar dust collector. 2005. 32

[144] Wikipedia contributors. List of earth observation satellites — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=List_of_Earth_observation_satellites, 2018. [Online; accessed 15-June-2018]. 1

[145] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. In *Advances in Neural Information Processing Systems*, pages 82–90, 2016. 64

[146] Jianxiong Xiao, Krista A. Ehinger, James Hays, Antonio Torralba, and Aude Oliva. SUN Database: Exploring a large collection of scene categories. *International Journal of Computer Vision (IJCV)*, 2014. 7, 14

[147] Xuehan Xiong and Fernando Torre. Supervised descent method and its applications to face alignment. In *Computer Vision and Pattern Recognition (CVPR)*, pages 532–539, 2013. 64

[148] Xinchen Yan, Jimei Yang, Kihyuk Sohn, and Honglak Lee. Attribute2Image: Conditional image generation from visual attributes. In *European Conference on Computer Vision (ECCV)*. 2016. 60, 76

[149] A. Yu and K. Grauman. Fine-grained visual comparisons with local learning. In *Computer Vision and Pattern Recognition (CVPR)*, June 2014. 73

[150] Fisher Yu, Wenqi Xian, Yingying Chen, Fangchen Liu, Mike Liao, Vashisht Madhavan, and Trevor Darrell. BDD100K: A diverse driving video database with scalable annotation tooling. *arXiv preprint arXiv:1805.04687*, 2018. 59

[151] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision (ECCV)*, pages 818–833. Springer, 2014. 12

[152] Haoqi Zhang, Eric Horvitz, and David C Parkes. Automated workflow synthesis. In *AAAI Conference on Artificial Intelligence*, 2013. 57

[153] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization. In *European Conference on Computer Vision (ECCV)*, pages 649–666. Springer, 2016. 60

[154] Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network. In *International Conference on Learning Representations (ICLR)*, 2017. 63

[155] Shuai Zheng, Ming-Ming Cheng, Jonathan Warrell, Paul Sturgess, Vibhav Vineet, Carsten Rother, and Philip HS Torr. Dense semantic image segmentation with objects and attributes. In *Computer Vision and Pattern Recognition (CVPR)*, pages 3214–3221. IEEE, 2014. 12

[156] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip HS Torr. Conditional random fields as recurrent neural networks. In *International Conference on Computer Vision (ICCV)*, pages 1529–1537, 2015. 31

[157] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using Places database. In *International Conference on Neural Information Processing Systems (NIPS)*, 2014. 7, 14, 43

[158] Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A. Efros. View synthesis by appearance flow. In *European Conference on Computer Vision (ECCV)*, pages 286–301. Springer, 2016. 60

[159] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A. Efros. Generative visual manipulation on the natural image manifold. In *European Conference on Computer Vision (ECCV)*, 2016. 84